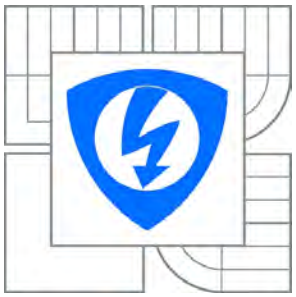
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

ZPRACOVÁNÍ OBRAZU V SYSTÉMU ANDROID - ODEČET HODNOTY ELEKTROMĚRU

IMAGE PROCESSING USING ANDROID DEVICE - ELECTRICITY METER VALUE RECOGNITION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. JIŘÍ SLIŽ

AUTHOR

VEDOUCÍ PRÁCE — Ing. PETER HONEC, Ph.D.

SUPERVISOR

BRNO 2015

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student: Bc. Jiří Sliž
Ročník: 2

ID: 134608
Akademický rok: 2014/15

NÁZEV TÉMATU:

Zpracování obrazu v systému Android - odečet hodnoty elektroměru

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce bude vytvoření programu pro zařízení Android, který bude využívat integrovanou kameru/fotoaparát k získávání obrazu. Práce bude rozdělena do následujících bodů:
1. Proveďte rešerši Android vývojového prostředí a algoritmů zpracování obrazu souvisejích s řešenou problematikou.
2. Vytvořte GUI a interface pro fotoaparát.
3. Navrhněte algoritmy pro zpracování obrazu - vyhodnocení snímku elektroměru, detekce pozic číslic včetně desetiného místa a odečtení hodnoty měřidla.
4. Implementujte algoritmy do zařízení.
5. Ověřte funkcionalitu a spolehlivost detekce a rozpoznání na reálných snímcích. Zaměřte se na nejčastější typy elektroměrů.

DOPORUČENÁ LITERATURA:

Hlaváč, Šonka, Počítačové vidění.
Šonka, Hlávač, Boyle - IMAGE PROCESSING, ANALYSIS, AND MACHINE VISION,

Termín zadání: 9. 2. 2015 ***Termín odevzdání:*** 18.5.2015

Vedoucí práce: Ing. Peter Honec, Ph.D.
Konzultanti diplomové práce:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Cílem práce je návrh aplikace pro mobilní zařízení s operačním systémem Android. Tato aplikace umožní snímání obrazu pomocí kamery zařízení a zpracování obrazu s podporou knihovny OpenCV. Účelem této aplikace je automatické odečtení hodnoty počítadla analogového elektroměru. Text obsahuje popis těchto elektroměrů. Následuje charakteristika systému Android a na tuto část přímo navazuje návrh samotné aplikace. Dalším celkem je návrh algoritmů zpracování obrazu, jejich testování a implementace do připravené Android aplikace.

KLÍČOVÁ SLOVA

Elektroměr, Android, OpenCV, Zpracování obrazu, OCR

ABSTRACT

The aim of the work is to design an application for mobile devices with the Android operating system. This application allows image capturing with a camera and image processing with the support of the OpenCV library. The purpose of this application is automatic value recognition of the analog electrometer. The text contains a description of the analog electrometers. The following is characteristic of Android operating system, and this part is directly connected to a draft of the application itself. Next part contains the image processing algorithms, testing and implementation into Android application.

KEYWORDS

Electrometer, Anrdoid, OpenCV, Image processing, OCR

SLIŽ, Jiří *ZPRACOVÁNÍ OBRAZU V SYSTÉMU ANDROID - ODEČET HODNOTY ELEKTROMĚRŮ*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Fakulta elektrotechniky a komunikačních technologií, 2014. 72 s. Vedoucí práce byl Ing. Peter Honec, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „ZPRACOVÁNÍ OBRAZU V SYSTÉMU ANDROID - ODEČET HODNOTY ELEKTROMĚRŮ“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji rodičům za gramatickou korekturu. Vedoucímu práce za nasměrování mé tvorby správným směrem.

Brno ....

(podpis autora)

OBSAH

Úvod **11**

1 Elektroměry **12**
1.1 Dělení elektroměrů . . . 12
1.2 Analogové elektroměry . . . 12
1.2.1 Princip funkce . . . 12
1.2.2 Typy analogových elektroměrů . . . 14

2 Android OS **15**
2.1 Historie . . . 15
2.1.1 Verze Android OS . . . 15
2.2 Architektura Android OS . . . 16
2.3 Vývojové nástroje . . . 17
2.3.1 Eclipse IDE . . . 18
2.4 Životní cyklus aplikace Android . . . 18
2.5 Intents . . . 20
2.6 Servises . . . 20
2.7 Návrh uživatelské aplikace . . . 21
2.7.1 Návrh GUI . . . 21
2.7.2 Struktura aplikace . . . 23
2.8 Popis aplikace Elektroměry . . . 24
2.8.1 GUI . . . 25
2.8.2 Aktivity . . . 26
2.9 Souhrn aplikace Eletroměry . . . 29

3 OpenCv **30**
3.1 Struktura . . . 30

4 Software pro testování **31**

5 Zpracování obrazu **32**
5.1 Předzpracování . . . 32
5.1.1 Jasová korekce . . . 32
5.1.2 Šum v obrazu . . . 33
5.1.3 Potlačení šumu . . . 35
5.1.4 Matematická morfologie . . . 36
5.1.5 Houghova tronsformace . . . 37
5.1.6 Automatická rotace . . . 37

5.2 Segmentace 39
5.2.1 Segmentace číselníku verze 1 39
5.2.2 Segmentace číselníku verze 2 43
5.2.3 „Real time“ detekce číselníku 46
5.2.4 Úspěšnost lokalizace číselníku 47
5.2.5 Segmentace číslic 47
5.2.6 Přesnost segmentace znaků 53
5.3 OCR 53
5.3.1 Geometrické deskriptory 53
5.3.2 Srovnávání významných bodů 56
5.3.3 Umělé neuronové sítě 59
5.3.4 Srovnání metod OCR 63
5.4 Vyhodnocení 63

6 Závěr **64**

Literatura **66**

Seznam symbolů, veličin a zkratek **69**

Seznam příloh **70**

A Přiložené soubory **71**

B Screen aplikace **72**

SEZNAM OBRÁZKŮ

1.1 Klasické provedení jednofázového elektroměru [7] 13
1.2 Panely a) jedno-tarifového [28] a b) dvou-tarifového [27] elektroměru. 14
2.1 Poddíl jednotlivých verzí Android na trhu 16
2.2 Grafické znázornění arhitektury Android OS [19] 17
2.3 Životní cyklus aplikace Android [9] 19
2.4 Diagram obrazovek grafického rozhraní 21
2.5 Projektová struktura aplikace Elektroměry 24
2.6 Návrh obrazovky hlavní nabídky. 25
2.7 Návrh obrazovek. (a) Zobrazení obrazu, b) seznam naměřených hodnot. 26
3.1 Součásti OpenCV [5] 30
4.1 Testovací software 31
5.1 Equalizace histogramu: a) originalní šedotónový obraz, b) histogram originálu, c) ekvalizovaný obraz, d) histogram ekvalizovaného histogramu. 34
5.2 Příklady jednoduchých strukturálních elementů. 36
5.3 Příklad Houghovy transformace. (a) Úsečka v prostoru obrazu (x, y), (b) prostor příznaků (θ, r). [16] 38
5.4 Automatická korekce rotace. a) Originální obraz, b) nalezené čáry v obrazu hran, c) otočený obraz na základě natočení dominantních čar. 38
5.5 Počítadla běžných elektroměrů. 39
5.6 Příklad prahování dvou obrazů. (a, d) Originální obrazy ve stupních šedě, (b, e) jednoduché prahování, (c, f) adaptivní prahování. 40
5.7 Úprava obrazu pro tvorbu masky pozadí. (a) šedotónový obraz elektroměru, (b) odstraněné detaily morfologickým uzavřením, (c) výsledek prahování. 41
5.8 Obraz vertikálních hran použitím Sobel operátoru. (a) šedotónový obraz elektroměru, (b) obraz vertikálních hran, (c) aplikovaná binární maska a odstranění nízké hladiny šumu, (d) prahovaný obraz hran, (e) odstraněny příliž velké objekty. 42
5.9 Demonstrace lokalizace číselníku z obrazu hran. (a) Vyznačené oblasti řádků potenciálních textů, (b) ohraničení možných číselníků (modře) a oblast číselníku (zelená). 43
5.10 Příklady masek příznaků typu Haar.[5] 44
5.11 Blokobé schéma struktury kaskádového klasifikátoru. 45
5.12 (a) Vnitřní mechanizmus počítadla elektroměru [15], (b) vzor číslic elektroměru. 48

5.13 Příklad prahování číselníku metoda 1. (a, e) Šedotónový obraz, (b, f) zaostřený obraz, (c, g) filtrace šumu, (d, h) adaptivní prahování. . . . 49
5.14 Příklad prahování číselníku metoda 2. (a, e) Šedotónový obraz, (b, f) adaptivní prahování, (c, g) odstranění malých objektů, (d, h) mediánový filtr. 50
5.15 Demonstrace odstranění okrajů a červeného rámečku. (a, e) Prahovaný obraz, (b, f) horizontální projekce (zvetšeno), (c, g) červená maska, (d, h) výsledek. 50
5.16 Segmentace číslic - vertikální projekce. (a) Binární předzpracovaný obraz, (b) vyplněné oblasti - šedě označena střední třetina, (c) vertikální projekce, (d) nalezené číslice, (e) normalizace šířky oblastí. . . . 51
5.17 Segmentace číslic - vertikální projekce. (a) Binární předzpracovaný obraz, (b) vyplněné oblasti - šedě označena střední třetina, (c) vertikální projekce, (d) nalezené číslice, (e) normalizace šířky oblastí, (f) odstraněny překrývající se oblasti. 52
5.18 Rozhodovací strom pro rozčlenění znaků na menší skupiny dle počtu objektů a otvorů. 56
5.19 Detekce významných bodů. (a) Vzory číslic elektroměru, (b) nalezené významné body metodou FAST. 57
5.20 Rozpoznání číslic pomocí srovnávání se vzory. (a) Vstupní obraz, (b) Segmentované symboly, (c-h) nejlepší nalezené shody se vzory jednotlivých číslic (vlevo test, vpravo vzor), (i-n) rozpoznané vzory. 58
5.21 Model neuronu. 60
5.22 Topologie dopředné vícevrstvé neuronové sítě perceptronů. 60
5.23 Příklady tréningových vzorů ANN. (a) Uměle vytvořené vzory, (b) vzory extrahované ze snímků elektroměrů. 61
B.1 Screeny aplikace. 72

SEZNAM TABULEK

2.1 Verze Android OS 16
5.1 Výsledky testů detekce číselníku elektroměru. 47
5.2 Výsledky testů detekce číselníku elektroměru. 53
5.3 Výsledky testů klasifikace s využitím geometrických deskriptorů. . . . 56
5.4 Výsledky testů klasifikace s využitím srovnání významných bodů. . . 59
5.5 Výsledky testů ANN pro 1 a 2 vnitřní vrstvy a rozsah počtu vnitřních neuronů 20-100. 62
5.6 Souhrn výsledků testů jednotlivých metod zpracování. 63

ÚVOD

Systém Android je v dnešní době jedním z nejrozšířenějších operačních systémů pro mobilní zařízení, která již disponují výkonným CPU a praktickými periferiemi, mezi něž patří řada senzorů a také kamera. Zařízení známé jako „Smart phone“ vlastní stále více lidí, proto se hledají další praktické aplikace těchto přístrojů. Vzhledem k narůstajícímu výkonu smartphonů je možné provádět efektivně i výpočetně náročnější úlohy, jakou je také zpracování obrazu. Cílem práce je tvorba mobilní aplikace pro systém Android umožňující automatické rozpoznání číslic a následné odečtení hodnoty na počítadlech analogových elektroměrů. K tomuto účelu se mobilní zařízení výborně hodí, protože dnes již standardně obsahuje kameru, rychlé CPU, připojení k internetu a v neposlední řadě je velmi rozšířeno mezi uživateli. V práci jsou dva hlavní tématické celky, prvním je popis systému Android a návrh grafického rozhraní pro danou úlohu. Druhým je návrh metod lokalizace číselníku, segmentace číslic a následně jejich rozpoznávání a odečet z obrazu elektroměru snímaného kamerou mobilního zařízení. Vypracován bude i popis analogových elektroměrů. Výsledkem bude plnohodnotná aplikace sloužící k odečtení hodnoty na analogovém elektroměru pouze ze snímku z kamery zařízení.

1 ELEKTROMĚRY

Elektroměr je měřicí přístroj určující velikost elektrické práce, tedy spotřeby elektrické energie na daném spotřebiči (domácnost).V současné době se pro měření spotřeby elektrické energie využívají různé druhy elektroměrů. Hlavní rozdělení elektroměrů je na analogové a digitální. Starší, ale stále ještě rozšířené, analogové elektroměry využívají převod elektrické energie na mechanickou, která působí na mechanické počítadlo elektroměru zobrazující spotřebu. Novější digitální typy využívají převodu napětí na číslo (ADC) a naměřené hodnoty ukládají do paměti mikropočítače, spotřeba se zobrazuje obvykle na LCD displejích.

1.1 Dělení elektroměrů

Rozdělení elektroměrů může být provedeno z různých hledisek.
Dělení podle počtu fází:

- Jednofázové
  - S jedním tarifem
  - S více tarify (obvykle se dvěma)
- Třífázové

Dělení podle způsobu měření:

- Analogové elektroměry
- Digitální elektroměry

1.2 Analogové elektroměry

Analogové elektroměry jsou stále hojně používané pro jejich životnost a možná také proto, že nevyvstala potřeba je měnit za nové digitální přístroje. Jejich odečítání je ovšem náročnější, protože pracovník musí přijít k zákazníkovi a s využitím vlastních smyslů opsat naměřené hodnoty. Novější digitální přístroje disponují např. bezdrátovým odečtem hodnoty.

1.2.1 Princip funkce

Analogové elektroměry pracují prakticky výhradně na principu indukčního měřicího přístroje. Náčrt vnitřního uspořádání elektroměru je znázorněn na obrázku 1.1. Přístroj se skládá z dvou elektromagnetů a rotační části tvořené hliníkovým diskem. Oba elektromagnety jsou tvořeny cívkou s jádrem tvaru E. První cívka (dolní) je proudová, obsahuje malý počet závitů vodičem o větším průměru. Tato cívka je

rozdělena na dvě poloviny, které jsou navinuty na protějších sloupcích jádra. Druhá cívka (horní) je napěťová, ta obsahuje velký počet závitů vodičem s malým průměrem. Napěťová cívka je navinuta na středním sloupku horního jádra. Ve vzduchové mezeře mezi oběma cívkami se otáčí hliníkový kotouč. Otáčky kotouče pomocí šnekového převodu pohybují otočnými číslicemi na číselníku elektroměru. Kotouč se také otáčí v magnetickém poli permanentního magnetu, který zajišťuje brzdící moment. [7]

Celý přístroj tedy funguje jako motor, jehož otáčky jsou úměrné odebíranému výkonu všech spotřebičů v měřeném úseku. Střední hodnota pohybového momentu je úměrná součinu magnetických toků obou cívek a sinu jejich fázového posunu. Pro pohybový moment rotoru platí vztah 1.1.

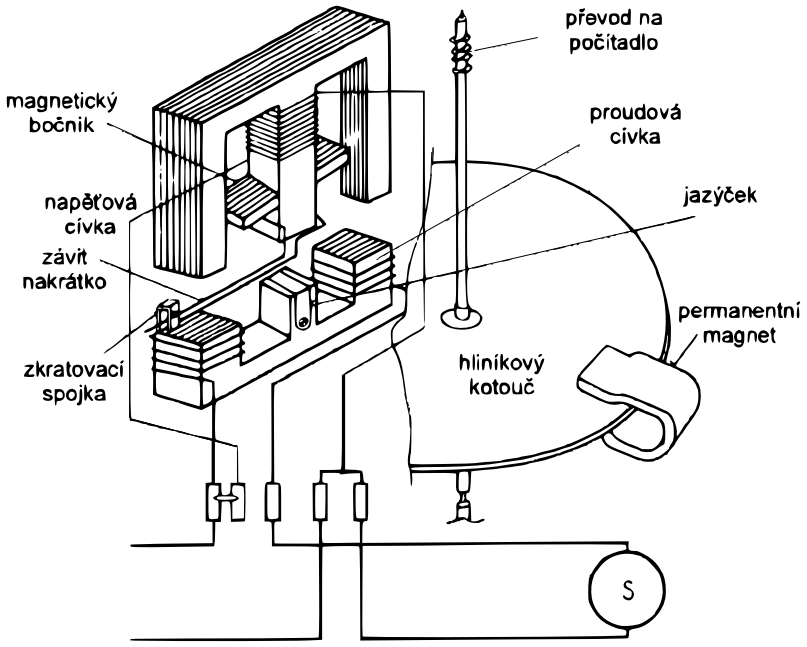
$$M_p = k\omega\Phi_U\Phi_I \sin\Psi \tag{1.1}$$

k - konstanta

ω - úhlová frekvence ($\omega = 2\pi f$)

Φ_U, Φ_I - amplitudy magnetických toků obou cívek

Ψ - fázový posun napěťové a proudové cívky.



Obr. 1.1: Klasické provedení jednofázového elektroměru [7]

1.2.2 Typy analogových elektroměrů

Existuje celá řada typů jednofázových analogových elektroměrů, lišících se převážně grafickým zpracování. Některé elektroměry jsou schopny měřit více tarifů (obvykle dva), ty mají dvě počítadla, pro každý tarif jedno.
Na obrázku 1.2 je zobrazen čelní panel jedno a dvou tarifového analogového elektroměru a popis důležitých částí. Obrázek 1.2 a) zobrazuje čelní panel jedno-tarifového elektroměru:

1. Mechanický číselník zobrazuje spotřebu elektrické energie v jednotce (2).
2. Jednotka zobrazované hodnoty spotřeby. V tomto případě [kWh], což je u elektroměrů nejrozšířenější.
3. Typové označení měřidla.
4. Výrobní číslo elektroměru.
5. Rotor elektroměru indikující odběr elektrické energie.

Na obrázku 1.2 b) je čelní panel dvou-tarifového elektroměru:

1. Mechanický číselník, zobrazuje spotřebu elektrické energie v jednotce (6). V tomto případě jsou zde číselníky dva, pro každý tarif jeden.
2. Signalizace aktivního tarifu.
3. Rotor elektroměru indikující odběr elektrické energie.
4. Typové označení měřidla.
5. Výrobní číslo elektroměru.
6. Jednotka zobrazované hodnoty spotřeby.

(a) (b)

Obr. 1.2: Panely a) jedno-tarifového [28] a b) dvou-tarifového [27] elektroměru.

2 ANDROID OS

Android je Operační Systém (OS) s linuxovým jádrem navržený zejména pro mobilní zařízení. Je vyvíjen společností Google a uskupením společností Open Handset Alliance (OAH). Tento OS je tvořen jako otevřená platforma, která odděluje software od hardware, na kterém běží. Tyto vlastnosti umožňující používání jedné aplikace na velkém množství různých zařízení samozřejmě lákají jak uživatele, tak vývojáře.

2.1 Historie

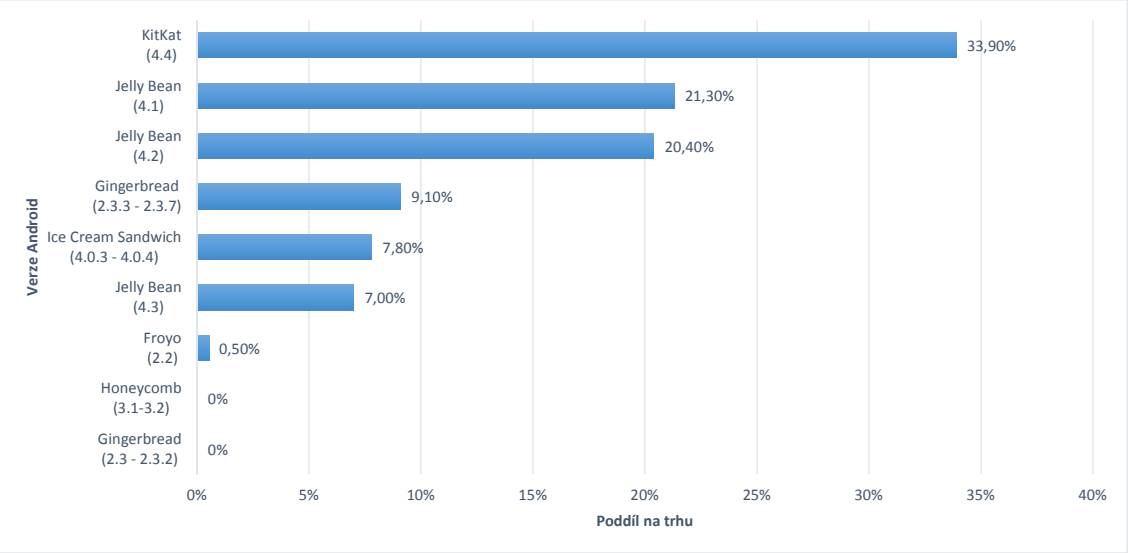
Android OS vytvořila skupina vývojářů z Kalifornie (Andy Rubin, Rich Miner, Nick Sears, Chris White) v roce 2003. Původním záměrem byla tvorba nového operačního systému pro digitální kamery, ovšem záměr se změnil na vytvoření OS konkurující soudobým systémům mobilních zařízení Symbian a Windows Mobile. Systém Android byl stále držen v tajnosti, zveřejnili jen informaci o vývoji nového systému pro chytré telefony.[2] V roce 2005 společnost Google koupila tento OS. První beta verze systému Android byla zveřejněna v listopadu 2007. V témže roku bylo vytvořeno uskupení společností OAH a Android se stal oficiálně open-source (Otevřený software). V roce 2008 byla zveřejněna první verze vývojového balíčku pro Android (Android SDK 1.0). Následujícího roku vyšlo několik verzí systému Android (od v1.5 do v2.1) a zvětšoval se počet mobilních zařízení využívajících tento OS. Další rok (2010) je Android druhým nejprodávanějším OS mobilních zařízeních hned za Blackberry. [9] Následující roky se Android OS stává naprostým leaderem v oblasti OS pro mobilní platformy. Koncem roku 2012 byl Android v 75 % mobilních zařízení podle výzkumu IDC (International Data Corporation).

2.1.1 Verze Android OS

V průběhu let vývoje již byla vydána celá řada verzí Android. Jednotlivé verze jsou označeny číslem verze a úrovně aplikace (API level - určeno pro vývojáře). Krom těchto identifikací jsou uvedeny názvy volené podle sladkostí (Cupkake, Jelly Bean, KitKat, ...). V tabulce 2.1 je seznam dosud vydaných verzí systému Android. Verzí Android OS je tedy celá řada, ovšem které verze se skutečně používají, o tom vypovídá diagram na obrázku 2.1. Z něj je také možné odhadnout, pro jaké verze je třeba aplikace vytvářet. Většině aplikací postačí zpětná kompatibilita do verze 2.3.3, tedy API level 10.

Android 1.0 (API level 1)	Android 3.1 Honeycomb (API level 12)
Android 1.1 (API level 2)	Android 3.2 Honeycomb (API level 13)
Android 1.5 Cupcake (API level 3)	Android 4.0–4.0.2 Ice Cream Sandwich (API level 14)
Android 1.6 Donut (API level 4)	Android 4.0.3–4.0.4 Ice Cream Sandwich (API level 15)
Android 2.0 Eclair (API level 5)	Android 4.1 Jelly Bean (API level 16)
Android 2.0.1 Eclair (API level 6)	Android 4.2 Jelly Bean (API level 17)
Android 2.1 Eclair (API level 7)	Android 4.3 Jelly Bean (API level 18)
Android 2.2–2.2.3 Froyo (API level 8)	Android 4.4 KitKat (API level 19)
Android 2.3–2.3.2 Gingerbread (API level 9)	Android 4.4 KitKat with wearable extensions (API level 20)
Android 2.3.3–2.3.7 Gingerbread (API level 10)	Android 5.0 Lollipop (API level 21)
Android 3.0 Honeycomb (API level 11)	

Tab. 2.1: Verze Android OS

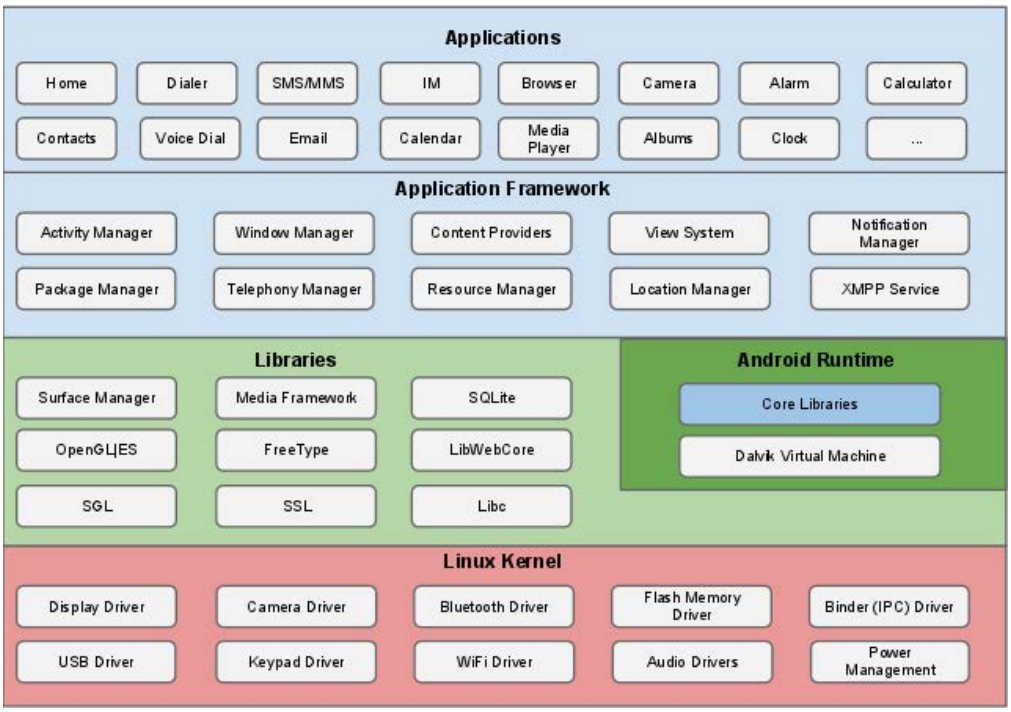


Obr. 2.1: Poddíl jednotlivých verzí Android na trhu

2.2 Architektura Android OS

Systém android se skládá z několika vrstev, které oddělují jednotlivé úrovně software. Hlavní vrstvy systému jsou čtyři, znázorněny jsou na obrázku 2.2. Nejnižší vrstvou, na obrázku 2.2 úplně dole, je jádro systému Linux. To podporuje multitasking, provádí správu paměti a také obsahuje ovladače hardware (např.: displej, USB, kamera, bluetooth, wifi, flash paměť, audio). Další vrstva obsahuje řadu knihoven psaných v C/C++ používaných jak systémem, tak aplikacemi. Přístup k nim zajišťuje Android Application Framework. Mezi tyto knihovny patří OpenGL ES a SDL (Simple DirectMedia Layer) pro tvorbu počítačové grafiky. Knihovna Surface manager vytváří uživatelské rozhraní systému. Také je k dispozici knihovna SQLite pro používání

SQL (Structured Query Language) databází. V této části se také nachází blok Android runtime, kde jsou obsaženy knihovny jádra (Core Libraries) a virtuální stroj Dalvik (Dalvik virtual machine). Dalvik je speciálně navržen pro systém android. Jedná se o podobný systém jako u Java VM (Virtual Machine), která zajišťuje chod Java aplikací na prakticky všech možných platformách. Dalvik je zaměřen pouze na mobilní zařízení, ovšem princip je stejný. To zajistí efektivnější práci s periferiemi mobilních zařízení a důsledkem je mimo jiné nižší spotřeba baterie. Následujícím celkem architektury je Aplication framework, který obsahuje nástroje pro správu aktivit (aplikací), zdrojů a grafických prvků a další. Nejvyšší vrstvou jsou samotné aplikace, které obstarávají funkce mobilního zařízení (sms, volání, hodiny/budík, kontakty, emaily, obrázky, ...). [9]



Obr. 2.2: Grafické znázornění arhitektury Android OS [19]

2.3 Vývojové nástroje

Android je zcela otevřenou platformou, k dispozici je pro vývojáře vývojový kit Android SDK (Software Development Kit). Primárně se vytváří Android aplikace v jazyku Java, ovšem je možnost použít jiných programovacích jazyků jako C/C++, Python, nebo C#. Navíc výrobce poskytuje kompletní vývojové prostředí Android Studio, založené na IntelliJ IDEA. Vzhledem k existenci Android SDK je možné

používat i mnoho dalších vývojových prostředí. Jedním z nejrozšířenějších prostředí pro vývoj aplikací Android je Eclipse IDE (Integrated Development Environment).

2.3.1 Eclipse IDE

Pro tuto práci jsem zvolil k vývoji Android aplikací prostředí Eclipse IDE. Oproti Android Studio je zde jednodušší správa více projektů a také použití nativního rozhraní JNI (Java Native Interface) pro vkládání částí kódů v jazyku C/C++. Eclipse IDE je univerzálním vývojovým nástrojem primárně určeným k vývoji v jazyku Java, ale v současné době je možné použít jej pro vývoj ve většině dnes používaných programovacích jazyků za pomoci Eclipse pluginů (rozšiřujících balíčků). A právě díky pluginům je toto prostředí velmi praktické i při vývoji pro Android.

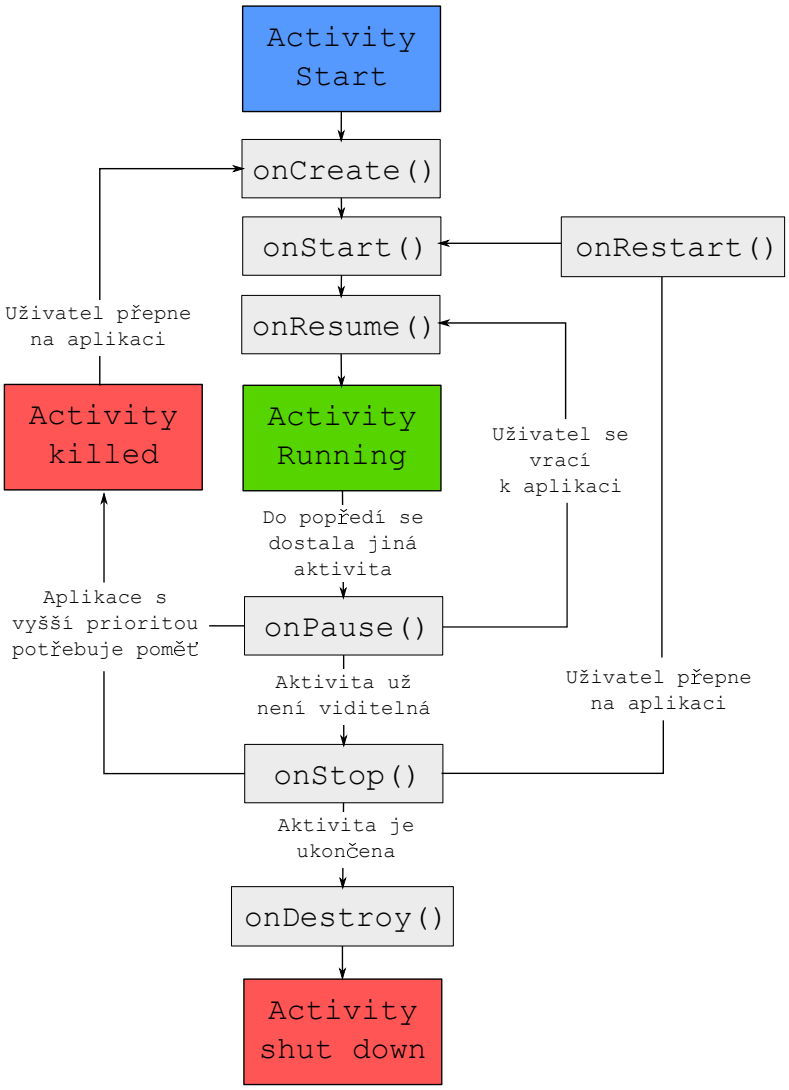
Konfigurace prostředí

Nastavení Eclipse prostředí tak, aby obsahovalo potřebné pluginy pro vývoj Android a umožnilo překlad C/C++ kódu a Java kódu současně, je poměrně pracná záležitost. Na internetu je množství různých návodů, které se nepatrně liší, a každý má své výhody a nevýhody. Naštěstí je možné se složité konfiguraci úplně vyhnout, a to díky vývojovému balíčku Tegra Android Development Pack, který poskytuje společnost NVIDIA zcela zdarma. Tento balíček nainstaluje všechny potřebné nástroje a provede konfiguraci. Kromě konfigurace Eclipse pro Android SDK je také přidána podpora nativního překladu C/C++ a instalována je navíc celá řada užitečných knihoven. Obsaženy jsou knihovny pro vývoj her, co je ale důležitější, také knihovu s funkcemi pro zpracování obrazu a videa OpenCV.[22]
V případě této práce byl pro vývoj uživatelské aplikace použit balíček Tegra Android Development Pack verze 3.0r. Tato verze zahrnuje Eclipse Kepler, Android SDK, Java development kit (jdk1.6.0_45) a knihovnu OpenCV 2.4.8.2.

2.4 Životní cyklus aplikace Android

Spouštění, rušení a další operace s aplikacemi zajišťuje v Android OS *Activity Manager*(Manažer aktivit). Aplikace mají více stavů než jen dvě (vypnuto/zapnuto). Je možné aplikaci pozastavit a spustit jinou, jakmile vyvstane potřeba návratu k předchozí aplikaci, jednoduše se obnoví z pozastaveného stavu. Tento proces byl navržen pro zrychlení uživatelského rozhraní. Vývojáři stačí ošetřit několik akcí pomocí připravených metod, aby tuto možnost implementoval. Není třeba psát program pro jednotlivé stavy aplikace, stačí pouze definovat, co se stane na přechodech mezi nimi.

Stavy aplikace a přechody mezi nimi jsou zobrazeny na obrázku 2.3.



Obr. 2.3: Životní cyklus aplikace Android [9]

Start aplikace následuje řada uživatelsky nastavitelných metod. Výsledkem startu aplikace je stav running (spuštěná). Proces spuštění aplikace je jedním z nejnáročnějších z hlediska výpočetního času, což ovlivní i výdrž baterie. To bylo hlavním důvodem vytvoření možnosti pozastavení aplikace, když není zrovna zobrazována namísto jejího ukončení. Uživatel se pravděpodobně k aplikaci ještě vrátí, proto jsou data načtena v paměti po určitou dobu.

Spuštěná aplikace (**Running**) je hlavní stav, kdy je aplikace zobrazena, běží a uživatel ji může využívat. Pro program to znamená, že veškeré vstupní akce (tlačítka, senzory, dotyk obrazovky,...) jsou řízeny touto aplikací. Platí, že je pouze

jedna spuštěná aplikace. Na systému android nemohou být ve stavu „running“ dvě grafické aplikace současně. Spuštěná aplikace má také vyšší prioritu na paměťový prostor a zdroje, aby mohla pracovat efektivně.

Pozastavená (**Paused**) aplikace je taková, která je stále zobrazena, ale není aktivní. Tento stav obvykle způsobí dialogová okna, či upozornění, které se objeví na displeji před aplikací. Pozastavené aplikace se blíží ke stavu zastaveno, ale stále disponují prioritou přidělení paměti a prostředků. Kdyby se aplikace při objevení dialogového okna zastavila úplně, mohlo by to způsobit špatné vykreslení na displeji.

Pokud aplikace není viditelná, ale stále je načtená v paměti, nastal stav zastaveno (**Stopped**). Zastavená aplikace může být znovu spuštěna, a to rychleji, než kdyby byla spouštěna ze zcela vypnutého stavu. Systém udržuje data aplikace, která byla zastavena, protože je pravděpodobné, že ji uživatel opět spustí.

Ukončená (**Destroyed**) aplikace se už nadále nenachází v paměti. V takovém případě Manažer aktivit rozhodl o uvolnění části paměti této aplikace. Před ukončením aplikace je možné provést potřebné akce uložení stavu, či bodu obnovy.
[9]

2.5 Intents

Intents (záměry) jsou v systému Anroid zprávy, které řídí spouštění a vypínání aktivit, služeb, nebo mohou být dokonce broadcast (pro všechny). Jedná se o asynchronní zprávy, to znamená žádné čekání na ně. Můžou být explicitní, nebo implicitní. Explicitní intent definuje přesně přijmací komponent. Implicitní intent specifikuje pouze žádaný typ příjemce. Příkladem implicitního intentu je žádost o otevření webové stránky. V takovém případě všechny aplikace, které jsou toho schopny, mohou tuto žádost vyřídit.
Pokud je více možností dokončení zadané žádosti, tak systém nabídne uživateli výběr formou dialogového okna a uživatel musí specifikovat, co bude použito.

2.6 Servises

Servis, neboli služba, je proces, který běží vždy na pozadí a nemá žádné uživatelské rozhraní. V podstatě mohou servisy provádět stejné úlohy jako aplikace, ovšem bez interakce s uživatelem. Servisy jsou velmi užitečné v případech, kdy je třeba provádět např. dlouhodobé akce (přehrávání hudby) a současně využívat jiné aplikace.

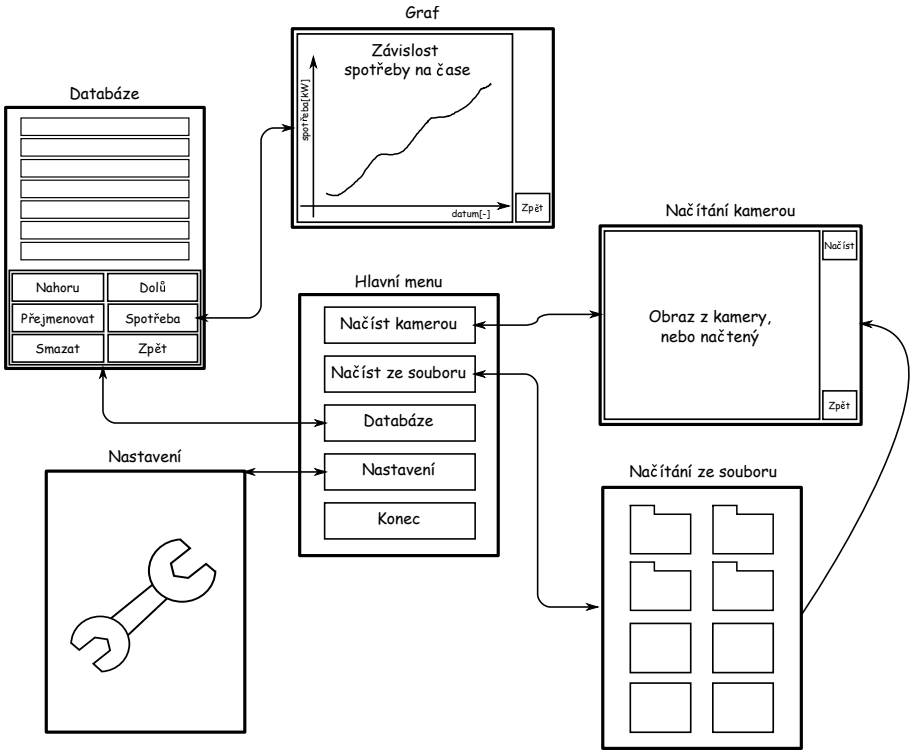
Podobně jako aplikace i servisy mají svůj životní cyklus. Ten je ovšem o hodně jednodušší. Jedná se pouze o stavy *Starting* - *Running* - *Destroyed*.

2.7 Návrh uživatelské aplikace

V této části bude popsán návrh uživatelského rozhraní pro Android aplikaci. Samotná aplikace je koncipována jako intuitivní a co možná nejjednodušší na ovládání. Nejdříve je třeba stanovit požadavky na aplikaci. Hlavní požadavky jsou dva. Prvním je možnost snímání elektroměru pomocí kamery, nebo načtení obrázku ze souboru. Druhým je databáze načtených hodnot, která umožní správu naměřených dat a jednoduché statistky spotřeby. Dalšími funkcemi nohou být nápověda, nastavení parametrů a změna algoritmu metody odečítání.

2.7.1 Návrh GUI

Grafické rozhraní aplikace obsahuje hlavní menu, ze kterého je možné provádět požadované akce. Na obrázku 2.4 je základní návrh jednotlivých obrazovek uživatelského rozhraní. Uprostřed je náčrt hlavní nabídky. Z ní je možno provézt požadovanou akci načítání a prohlížení dat, nebo nastavení aplikace.



Obr. 2.4: Diagram obrazovek grafického rozhraní

V prostředí Eclipse je k dispozici plugin grafický designer. GUI je definováno strukturálním popisem pomocí XML (Extensible Markup Language). Grafický návrh je tedy zcela oddělen od programu. To umožňuje flexibilitu při změnách vzhledu aplikace bez nutnosti zásahu do kódu.

Založení projektu

Prostředí Eclipse nabízí přehledný průvodce pro vytvoření projektu Android aplikace. Všechny potřebné složky a základní soubory jsou vygenerovány automaticky, podle zvolených parametrů:

1. **Název projektu** - je to název projektu, který právě vytváříme. Název vyžaduje Eclipse, který rozčleňuje jednotlivé projekty podle jména.
2. **Verze Android platformy** - musí být stanoveno pro kterou verzi bude aplikace určena. Pro zvolenou verzi musí být staženo příslušné SDK, které je možné stáhnout pomocí Eclipse pluginu. V našem případě byly potřebné platformy zvoleny již při instalaci. Jednotlivé verze jsou vypsány v tabulce 2.1.
3. **Název aplikace** - název výsledné aplikace, obvykle se používá hlavní jazyk aplikace popř. anglický název.
4. **Název balíčku** - balíčky jsou vyžadovány jazykem Java. Každá třída je umístěna v konkrétním balíčku a tyto pak určují viditelnost jednotlivých tříd mezi sebou. Název balíčku se skládá ze slov názvů oddělených tečkou. Jednotlivé názvy jsou na disku reprezentovány složkami vloženými do sebe. Často tyto názvy vyjadřují doménu vývojáře a název aplikace. V případě aplikace Elektroměry je název balíčku: sliz.elektromery.
5. **Aktivita** (Activity) - je reprezentována Java třídou. Tato třída obsahuje metody, které tvoří samotný program. Aktivit může být v celé aplikaci více, ale jedna musí být nastavena jako hlavní a ta bude také spuštěna po startu aplikace.
6. **Minimální verze Android platformy** - je potřeba stanovit kompatibilitu, tedy na jaké ze starších verzí Android bude zajištěna správná funkce aplikace. Pro aplikaci Elektroměry je stanovena kompatibilita od API level 8, což zahrnuje v podstatě všechny dnes používané verze (obr. 2.1).

[9]

2.7.2 Struktura aplikace

Projekt Android aplikace má předepsanou strukturu. Obsahuje některé soubory vždy a jiné jsou volitelné. V této části budou popsány důležité součásti projektu, hlavní jsou následující tři:

- Zdrojové kódy - program.
- Zdroje (resourses) - obsahují grafické návrhy, texty, obrázky, ...
- Projektové soubory - obsahují informace jak spojit vše dohromady, nastavení výsledné aplikace.

Zdrojové kódy

Jak již bylo zmíněno (kapitola 2.3), aplikace pro Android jsou nejčastěji vyvíjeny v jazyku Java. Je ovšem možné, díky nativnímu rozhraní, psát v jazyku C++. Zdrojové kódy Java, jsou zařazeny ve složce „src“, kde se nachází série v sobě vložených složek podle názvu balíčku a v poslední z nich jsou samotné zdrojové texty. V případě C++ je tomu jinak. Všechny zdrojové texty jsou umístěny ve složce „jni“ (Java Native Interface).
V případě aplikace Elektroměry jsou přítomny oba zmíněné druhy zdrojových kódů (Java i C++). Java kód zajišťuje funkcionalitu aplikace, tvoří jednotlivé aktivity, které ovládají uživatelské rozhraní a zprostředkovávají komunikaci s HW mobilního zařízení. V jazyku C++ jsou pouze psány metody pro zpracování obrazu, protože knihovna OpenCV je primárně psána v C/C++. Existuje sice Java implementace OpenCV, ale vzhledem k tomu, že testování algoritmů zpracování obrazu probíhá v PC v jazyku C++, je výhodnější použít právě tento jazyk i v Android aplikaci.

Zdroje

Zdroje zahrnují všechny soubory, které jsou v aplikace potřeba a nejsou přímo zdrojovými kódy. Mezi zdroje patří obrázky, včetně ikony aplikace, dále struktury v XML, které obsahují návrh GUI. Mezi další XML soubory patří seznam textů, které se v aplikaci vyskytují. Tento seznam je pojmenován „string.xml“ a může jich být v projektu více, pro překlad do jiných jazyků. Pokud aplikace používá externí soubory, je možné použít adresář „raw“, který svůj obsah připojí k aplikaci a jednotlivým položkám přiřadí symbolické adresy.

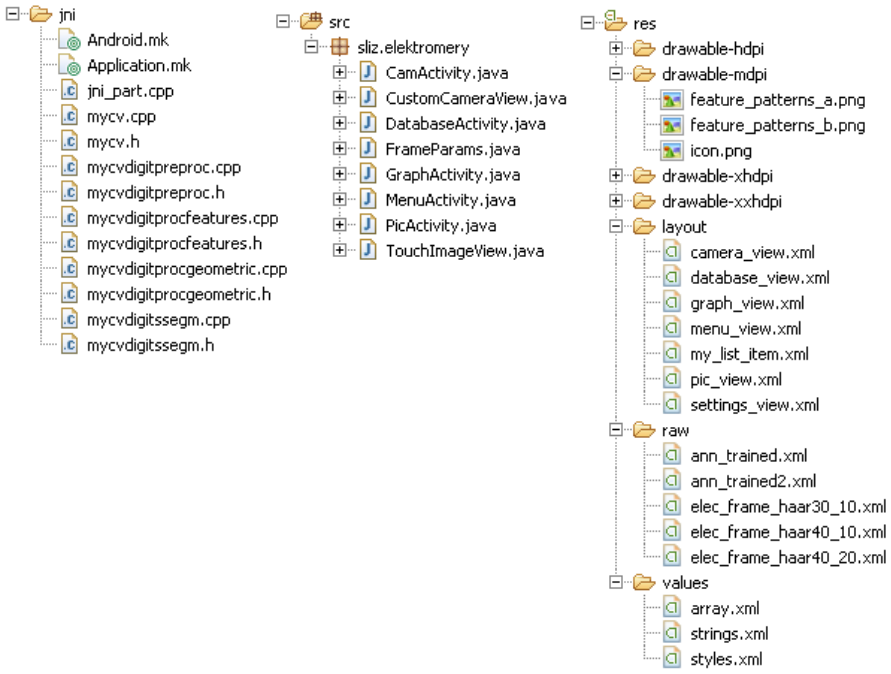
Projektové soubory

Poslední součástí jsou projektové soubory. Ty informují Eclipse o tom, jak má být ze zdrojových kódů a souborů složena aplikace. Stěžejním souborem je „AndroidManifest.xml“. Tento soubor obsahuje informace o tom, z čeho se aplikace skládá, jaká přístupové

práva bude mít, název aplikace, informaci o ikoně, styly a další vlastnosti. [9]

2.8 Popis aplikace Elektroměry

Celý projekt aplikace Elektroměry je přiložen v příloze práce. Někdy bude odkazováno na části kódů, které se nacházejí v projektu, ale z důvodů úspory místa nejsou všechny uvedeny v textu. Na obrázku 2.5 je projektová struktura aplikace Elektroměry.



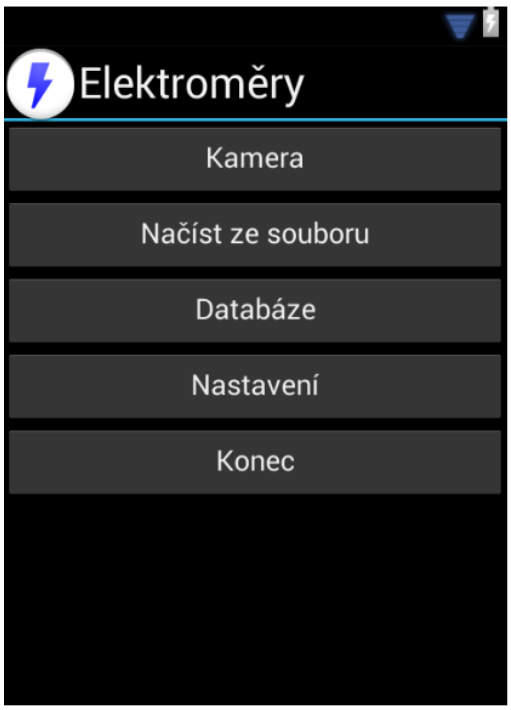
Obr. 2.5: Projektová struktura aplikace Elektroměry

V projektu je osm Java tříd, z nichž jedna představuje grafický prvek (Widget) určený pro práci s kamerou a zobrazování snímků (CustomCameraView.java), další je použitá jako struktura parametrů nalezeného rámečku číselníku (FrameParams.java). Jedna je převzatá z externího zdroje [24], zajišťuje funkce pro prohlížení obrázků (TouchImageView.java). Dalších pět tříd, vždy mají v názvu „Activity“, představují jednotlivé aktivity. Konkrétně jsou to hlavní nabídka (MenuActivity.java), načítání a prohlížení obrázku (PicActivity.java), ovládání kamery (CamActivity.java), správa načtených dat (DatabaseActivity.java) a vykreslení grafu (GraphActivity.java), kde je použita knihovna AndroidPlot [3]. Ve složce „jni“ jsou příslušné soubory typu „.mk“, zajišťující správný překlad nativního kódu a zdrojové kódy v jazyce C++. Jedná se o třídy určené pro zpracování obrazu číselníku s výjimkou souboru

„jni_part.cpp“, který obsahuje nativní metody volané z Java tříd. Dále se v projektu nacházejí zdroje ve složce „res“. Složka „layout“ obsahuje jednotlivé grafické návrhy uživatelských obrazovek. Přítomen je také seznam textů aplikace („strings.xml“). Ve složce „raw“ jsou umístěny soubory, které bude aplikace využívat, ty zahrnují naučenou neuronovou síť a kaskádový klasifikátor v podobě XML struktur. Složky začínající „drawable“ obsahují obrázky používané aplikací, kromě ikony jsou zde umístěny vzory číslic pro metody rozpoznávání vzorů (kapitoly 5.3.1 a 5.3.2).

2.8.1 GUI

Podle předchozího návrhu jednotlivých obrazovek grafického rozhraní (kapitola 2.7.1) byly s využitím modulu grafického designeru v prostředí Eclipse vytvořeny GUI jednotlivých obrazovek. Každý návrh je tvořen XML strukturou obsahující údaje o typu, rozmístění a parametrech grafických komponent. Na obrázku 2.6 je zobrazen vlevo grafický návrh hlavní nabídky aplikace a vpravo je příslušný XML kód. Hlavní nabídka obsahuje pouze pět tlačítek, která jsou pomocí prvku LinearLayout uspořádána pod sebou.




```
<LinearLayout
xmlns:android=
"http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"  >
    <Button
        android:id="@+id/btn_menu_open_cam"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/btn_menu_open_cam" />
    <Button
        android:id="@+id/btn_menu_open_pic"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/btn_menu_open_pic" />
    <Button
        android:id="@+id/btn_menu_database"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/btn_menu_database" />
    <Button
        android:id="@+id/btn_menu_settings"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/btn_menu_settings" />
    <Button
        android:id="@+id/btn_menu_end"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/btn_menu_end" />
</LinearLayout>
```

Obr. 2.6: Návrh obrazovky hlavní nabídky.

Na obrázku 2.7 jsou uvedeny další dva návrhy grafického rozhraní. Obr. (a) je

návrh rozhraní kamery. V případě zobrazení dat z kamery byla vytvořena třída CustomCameraView, která zajišťuje řízení i zobrazování dat z kamery. Struktura ostatních návrhů je patrná ze screenshotů aplikace uvedených v příloze. Obrazovka pro zobrazení načteného obrázku zobrazuje pouze statický obraz pomocí prvku TouchImageView [24].Na obrázku 2.7 (b) je zobrazen návrh obrazovky výčtu naměřených hodnot. Pro výčet hodnot je použit prvek ListView, který je určen pro zobrazování seznamu hodnot. Pod seznamem se nachází tlačítka na provedení potřebných úkonů s vybranou položkou v seznamu.

(a) (b)

Obr. 2.7: Návrh obrazovek. (a) Zobrazení obrazu, b) seznam naměřených hodnot.

2.8.2 Aktivity

Grafický návrh je hotov, ale k tomu, aby aplikace byla funkční, je třeba vytvořit třídy tvořící aktivity. V tomto projektu je pro každou navrženou obrazovku vytvořena jedna aktivita (Java třída), která zajišťuje funkčnost dané obrazovky. Jednotlivé aktivity je možné podle potřeby přepínat a tak přecházet z jedné obrazovky na druhou. Každá aktivita obsahuje metodu *onCreate*, která je volána po vytvoření třídy. V této metodě se obvykle zobrazí příslušná obrazovka metodou *setContentView*, která na základě id vykreslí obrazovku příslušného grafického návrhu. Jakmile je toto provedeno, je možné zpracovávat události vzniklé interakcí uživatele s danou obrazovkou (např. stisk tlačítka).

MenuActivity

První třída (aktivita), která se spustí při startu aplikace Elektroměry, má název *MenuActivity*. Nastavení aplikace při spuštění se provádí v konfiguračním souboru „AndroidManifest.xml“. Třída *MenuActivity* obsahuje pouze ošetření stisku tlačítek hlavní nabídky a zobrazení nastavení. Ošetření stisku tlačítka se provádí pomocí třídy *onClickListener*, která obsahuje metodu *onClik*, která je volána po stisku tlačítka. Zdrojový text nastavení akce prvnímu tlačítku v hlavní nabídce je níže (Kód 2.1). Tlačítku je přidělen *onClick* listener („hlídač akce“), který provede spuštění aktivity CamActivity a ukončí hlavní nabídku, čímž se přepne obrazovka na snímání kamerou. Ostatní tlačítka jsou analogicky přiřazena k akcím změny příslušné obrazovky, poslední tlačítko ukončí celý program.

```
// Bnt camera
Button btnCam = (Button)findViewById(R.id.btn_menu_open_cam);
btnCam.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intnscnd = new Intent(MenuActivity.this, CamActivity.class);
        startActivity(intnscnd);
        MenuActivity.this.finish();

    }
}
);
```

Kód 2.1: Nastavení akce při stisku tlačítka

CamActivity

Pro odečtení hodnoty z elektroměru bude zapotřebí pouze namířit kameru mobilního zařízení na čelní panel měřicího přístroje. Jakmile aplikace rozpozná číselník/číselníky, pořídí automaticky snímek, který bude zpracován a zobrazí výsledek. Snímání scény kamerou mobilního zařízení zajišťuje aktivita *CamActivity*. Tato třída implementuje *CvCameraViewListener2* obsažený ve třídě *CameraBridgeViewBase*, kterou poskytuje knihovna OpenCV pro Javu. To umožňuje třídě *CamActivity* zpracovávat video z kamery zařízení snímek po snímku. Výsledkem je „real time“ video zpracování videa.

Pro vykreslování obrazu z kamery a její interaktivní ovládání byla vytvořena třída *CustomCameraView*, která dědí *JavaCameraView*. Tato třída umožňuje načíst parametry kamery mobilního zařízení, na kterém aplikace běží. Mezi tyto parametry patří podporovaná rozlišení, efekty kamery, módy kamery a další. Třída také obsahuje *PictureCallback*, díky kterému je možné sejmout a uložit snímek v plném

rozlišení s využitím automatického zaostření.
Vrátíme se k třídě *CamActivity*. Byla reimplementovaná metoda pro stisk tlačítka zpět, které namísto vypnutí aplikace provede pomocí intentu (viz kapitola 2.5) návrat do hlavní nabídky. Tato metoda se nachází i v ostatních třídách implementujících aktivitu. Dále je připraveno kontextové menu pro nastavení parametrů kamery zařízení. Zjištění parametrů zajišťuje třída *CustomCameraView*. V GUI zobrazení kamery jsou k dispozici dvě tlačítka. Jedno pro návrat do hlavní nabídky a druhé slouží k načtení elektroměru. Návrh je znázorněn na obrázku 2.7 a).
Aby bylo možné zpracovávat video z kamery pomocí nativních funkcí v C/C++, musí být po spuštění aktivity *CamActivity* načtena nativní knihovna. To je provedeno pomocí *BaseLoaderCallback*, což je třída, kterou k tomuto účelu poskytuje OpenCV. Pro uložení aktuálního snímku kamery jsou k dispozici dvě proměnné typu Mat (matice), jedna *mRgba* pro práci s barevným obrazem a druhá *mGray* pro rychlejší zpracování šedotónového obrazu. Instance zmíněné třídy *CustomCameraView* se jmenuje *mOpenCvCameraView* která, jakmile je spuštěna (kamera snímá), vyvolá callback metodu (*onCameraViewStarted*) kde se potřebné proměnné pro zobrazení obrazu alokují. V metodě volané po ukončení snímání kamerou se příslušné proměnné zase uvolní.
Přístup ke každému snímku videa zajišťuje implementovaná třída *CvCameraViewListener2*, která poskytuje metodu *onCameraFrame*. Tato metoda je zavolána pokaždé, když kamera dokončí snímání jednoho snímku videa (asynchronní děj).

PicActivity

Zobrazení uloženého obrázku a samotné odečtení hodnoty bude provedeno v aktivitě *PicActivity*. Obdobně jako u třídy *CamActivity* je i zde využíváno nativního rozhraní pro volání C/C++ metod s podporou OpenCV. K načtení nativních metod je použit *BaseLoaderCallback*. Pokud se aktivita *PicActivity* spustí z hlavní nabídky, je nejprve otevřena standardní galerie obrázků, ze které uživatel vybere příslušný snímek elektroměru. V případě přechodu ze snímání kamerou se automaticky načte obrázek právě vyfoceného elektroměru. Následně je provedena detekce číselníku a zobrazen bude již oříznutý rámeček, kde budou označeny nalezené číslice. Pod obrázkem se nachází textové pole, kde je vypsáno rozpoznané číslo hodnoty číselníku. Pokud dojde k nepřesnému odečtení, má uživatel možnost hodnotu opravit. Ukončení odečítání se provede tlačítkem „Načíst“. Díky použité knihovně *TouchImageView* je možné obrázek přiblížit a jednoduše pomocí gest prohlížet.

DatabaseActivity

Tato aktivita obsahuje prvek *ListView*, který umožňuje práci se seznamy. Jednotlivé odečtené hodnoty jsou zobrazeny pod sebou a je možná úprava položek a jejich mazání. Seznam je spojen se souborem typu „csv“, jenž obsahuje naměřené hodnoty a příslušné datum odečtení. Název souboru identifikuje odečítaný elektroměr a může jich být uloženo více, pokud uživatel chce znát statistiky více domácností. Ovšem aktuálně načtený je vždy jeden soubor.

GraphActivity

K vykreslení naměřených hodnot spotřeby v závislosti na čase slouží *GraphActivity*. Zde je použita externí knihovna AndroidPlot [3] umožňující tvorbu grafu.

2.9 Souhrn aplikace Eletroměry

Aplikace je plně funkční, podle specifikací zadání a je možné ji používat. Nicméně nebyla dlouhodobě testována, takže není optimálně nastavena pro nepředpokládané podmínky a požadavky. Použití lze shrnout do následujících bodů:

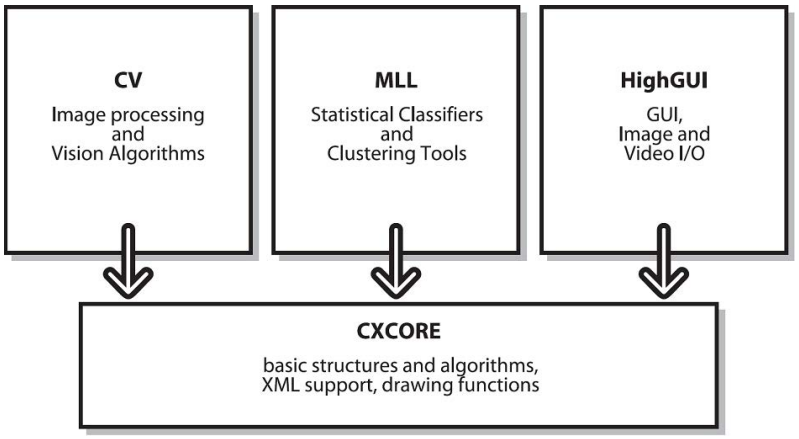
- Spuštění aplikace na mobilním zařízení Android.
- Výběr volby *kamera* a namíření zařízení na snímaný elektroměr.
- Nyní již uživatel pouze obdivuje aplikaci, která automaticky vyhledá číselník (číselníky) elektroměru a jakmile se tak stane, pořídí snímek, ze kterého rozpozná hodnotu spotřeby.
- Uživateli jsou pro kontrolu předloženy výsledky, a ty je možné uložit do databáze.

3 OPENCV

Pro zpracování obrazu je v projektu využíváno open-source knihovny OpenCV (Open Source Computer Vision Library), která obsahuje množství funkcí pro zpracování obrazu. Tato knihovna je původně psaná v jazyku C/C++, ale existují i možnosti, jak ji použít například v jazyku Java, Python, Matlab či C#. Knihovna je optimalizovaná na výkon a využití v aplikacích reálného času. Vývoj OpenCV začal ve společnosti Intel Research.

3.1 Struktura

Knihovna OpenCV se skládá z hlavních pěti celků. Čtyři z nich jsou graficky zobrazeny na obrázku 3.1. Blok **CV**(Computer Vision) obsahuje základní funkce zpracování obrazu a dále pokročilejší algoritmy počítačového vidění. **ML**(Machine Learning) strojové učení obsahuje implementace statistických klasifikátorů a shlukové analýzy. Další částí je **HighGUI** obsahující grafické rozhraní a vstupně výstupní interface pro obraz a video. **CXCORE** obsahuje potřebné datové struktury a rutiny k propojení jednotlivých komponent. Posledním prvkem, který se nenachází na obrázku 3.1, je **CvAux** obsahující jednak algoritmy rozpoznání obličeje s podporou vestavěných systémů (embeded) a také experimentální algoritmy segmentace. Poslední zmíněná část není úplně oficiální a tomu odpovídá také neúplná dokumentace. [5]



Obr. 3.1: Součásti OpenCV [5]

4 SOFTWARE PRO TESTOVÁNÍ

Android aplikace je praktická v případě potřeby mobility, ta ovšem vždy nevyvstává při vývoji, a z tohoto důvodu byl vytvořen jednoduchý program nazvaný *Electrometer testing* určený k testování algoritmů. Software je založen na QT frameworku a je tedy multiplatformní (Windows, Linux, Mac). Program má jednoduché grafické rozhraní, umožňující načítání obrázků ve složce, jejich přepínání a provádění testů. Screenshot programu je na obrázku 4.1.

Obr. 4.1: Testovací software

Vzhledem k použití nativního rozhraní v Android aplikaci je možné otestovaný algoritmus v C/C++ přímo zkopírovat do projektu aplikace a použít jej ve spolupráci s mobilním zařízením. Díky OpenCV je možné vykreslit rychle funkcí *imshow(...)* jednotlivé obrazy v průběhu zpracovávání. Postupně byly do programu přidávány další nástroje k usnadnění vývoje, zde je výčet: Otevření/Uložení obrazu včetně výřezů číselníků a číslic do samostatných složek, dále jsou k dispozici nástroje pro testování jednotlivých metod klasifikace, učení neuronové sítě. K tvorbě trénovacích dat je k dispozici dávkové zpracování obrazů.

5 ZPRACOVÁNÍ OBRAZU

V následujícím textu je popsáno zpracování obrazu s účelem odečtení hodnoty elektroměru. Nejprve jsou popsány metody předzpracování, následuje segmentace číslic číselníku z obrazu a poslední částí je rozpoznávání číslic. V práci je využita knihovna OpenCV (kapitola 3), která obsahuje množství funkcí pro zpracování obrazu. Teoretický popis algoritmů bude doplněn o popis příslušné funkce z této knihovny, pokud se v ní nachází. Na závěr každé kapitoly bude stručně popsána implementace zmíněných metod s odkazy na zdrojové kódy.
Třídy (C++) zpracování obrazu implementované v tomto projektu mají vždy předponu *mycv* (My Computer Vision), což je taky název hlavní třídy, obsahující řadu metod používaných dalšími třídami.

5.1 Předzpracování

Prvním krokem při zpracování obrazu je jeho předzpracování. Tato část obvykle zahrnuje potlačení šumu, odstranění zkreslení a případné zvýraznění, či naopak potlačení určitých objektů. Jsou to operace na nejnižší úrovni zpracování a jak vstupem, tak výstupem je obraz jasových hodnot.

5.1.1 Jasová korekce

Šedo-tónový obraz je vyjádřen jasovými úrovněmi v pravidelné mřížce. Četnosti jednotlivých jasových hodnot v obrazu jsou nejčastěji vyjadřovány pomocí histogramu. V ideálním případě je obraz dán pouze barvou objektů obsažených ve scéně, nicméně je také ovlivněn okolním osvětlením, odlesky, stíny a také šumem.

Histogram

Histogram zobrazuje četnosti jednotlivých jasových úrovní v obrazu, neboli distribuční funkci pravděpodobnosti výskytu pixelu dané jasové hodnoty. Příklad histogramu je zobrazen na obrázku 5.1 b). Z hodnot histogramu je možné odvodit další parametry, které se používají v dalším zpracování. Integrací histogramu vznikne tzv. kumulovaný histogram. Ten je tvořen monotónně rostoucí funkcí. Výpočet kumulovaného histogramu je možné provést podle vzorce 5.1.

$$h_k(k) = \sum_{i=0}^{k} h(i) \tag{5.1}$$

kde h - hodnoty histogramu, h_k - hodnoty kumulovaného histogramu a k - celkový počet jasových úrovní v obrazu.

Ekvalizace histogramu

Obraz pořízený při slabém, či příliš silném osvětlení bude mít značně nerovnoměrné rozložení hodnot histogramu. Takový obraz může být velmi matný, či příliš světlý. Ekvalizace histogramu zajistí rovnoměrné rozložení jednotlivých jasových složek v obrazu a zvýší kontrast. Postup je následující:

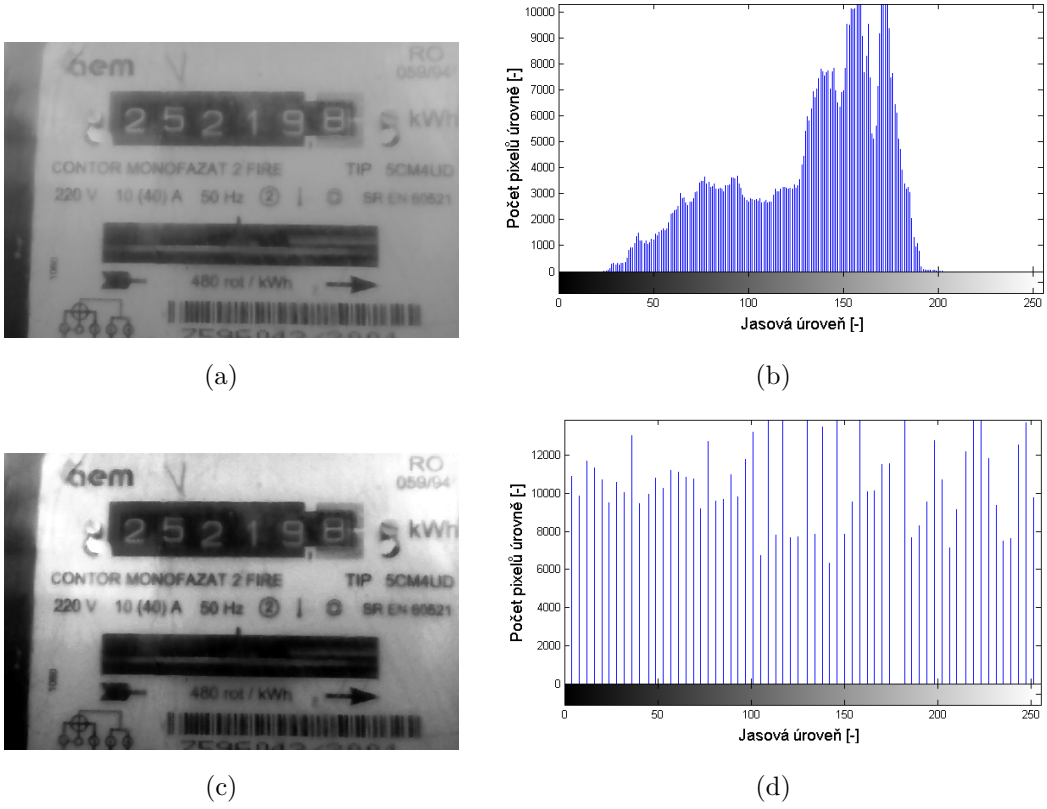
1. Výpočet histogramu obrazu
2. Výpočet kumulovaného histogramu
3. Normalizace kumulovaného histogramu na počet jasových úrovní
4. Jasová transformace vytvořenou funkcí

Na obrázku 5.1 je příklad ekvalizace snímku elektroměru. Horní dva obrázky představují originální snímek ve stupních šedi a jeho histogram. V dolní části obrázku 5.1 je výsledný ekvalizovaný obraz a jeho histogram. Je jasně vidět, že výsledný histogram má rovnoměrně rozložené hodnoty přes celý rozsah jasových úrovní.

Ekvalizaci histogramu je možné provést také na barevných obrazech. V takovém případě je důležité zvolit vhodný barevný model. Barevné modely, které neoddělují barevnou složku od jasové nejsou příliš vhodné, protože je proces pro každou barevnou složku nezávislý a některé složky tak mohou být příliš zvýrazněny, zatímco jiné potlačeny. Vhodnými barevnými modely jsou např.: YUV (Y - jasová složka, U a V - barevné složky), HSV/HSL (Hue - barevný tón, Saturation - sytost barvy, Value/Lightness/Luminance - jasová složka). U takového barevného modelu se provede ekvalizace pouze u jasové složky.
Opencv : K dispozici je funkce *equalizeHist(...)*.

5.1.2 Šum v obrazu

Šum v obrazu zpravidla nastává v důsledku nedokonalosti snímacího zařízení, při transmisi signálu, nebo v důsledku zpracování. Vzhledem k tomu, že šum je náhodný jev, popisuje se obvykle pomocí pravděpodobnostních charakteristik. Šum může být zcela nezávislý na obrazu, nebo naopak závislý. V případě nezávislého mluvíme o tzv. aditivním šumu, který je separovatelný od obrazové informace.[21]



Obr. 5.1: Equalizace histogramu: a) originalní šedotónový obraz, b) histogram originálu, c) ekvalizovaný obraz, d) histogram ekvalizovaného histogramu.

Idealizovaný šum, jenž má rovnoměrnou spektrální hustotu, se nazývá **bílý šum**. Všechny frekvenční složky šumu jsou přítomny a mají stejnou intenzitu. Bílý šum je možné použít jako model nejhorší možnosti zkreslení tohoto typu. Speciálním případem bílého šumu je **Gaussův šum**, jehož náhodná veličina má pravděpodobností rozdělení dáno Gaussovou křivkou:

$$p_x = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{5.2}$$

kde μ je střední hodnota a σ šířka křivky.

Kvantizační šum je dalším typem, který vzniká při omezování počtu jasových úrovní. Tento typ zkreslení může nastat již při pořízení obrazu, protože kamery vždy provádějí kvantování jasových úrovní, nebo při dalším zpracování. Může způsobit falešné kontury objektů a oblastí v obrazu.[21]

Impulzní šum se projevuje individuálně v každém pixelu obrazu, jejichž hodnota se pak výrazně liší od sousedních. Saturovaný impulzní šum, tedy s pouze dvěma hodnotami úrovní jasu, se nazývá salt-and-pepper (sůl a pepř).[21]

5.1.3 Potlačení šumu

Potlačení, či úplné odstranění šumu je možné pouze za předpokladu znalosti charakteru tohoto zkreslení. V obrazech pořízených CCD nebo obdobnou technologií se obvykle vyskytuje aditivní šum podobný Gaussovu typu. Odstranění šumu se provádí pomocí některé z metod filtrace obrazu.

Filtrace obrazu

Filtraci obrazu můžeme zařadit do lokálních a někdy globálních jasových transformací, kde hodnota pixelu výsledného obrazu závisí na lokálním okolí, respektive celém vstupním obrazu. Tento proces je možné provést pomocí operace konvoluce daného obrazu a tzv. konvoluční masky dle vztahu 5.3 (2D diskrétní konvoluce). V případě odstraňovaní se používají vyhlazovací masky, mezi které patří lokální statistiky (průměr, medián), gaussův filtr. Taková filtrace potlačuje vyšší frekvence, což zahrnuje nežádaný šum, ale také ostré přechody jako jsou hrany v obrazu.

$$(f * h)(x, y) = g(x, y) = \sum_{i=-\frac{m}{2}}^{\frac{m}{2}} \sum_{j=-\frac{n}{2}}^{\frac{n}{2}} h(x - i, y - j) \cdot f(i, j) \tag{5.3}$$

kde f je vstupní obraz, h je konvoluční maska, m a n jsou rozměry masky. Často se používá čtvercová maska s lichým počtem pixelů strany, aby mohl být počítaná bod ve středu masky.

Lineární filtrace

Odezva lineárního filtru je dána lineární kombinací masky a výřezu zpracovávaného obrazu. K tomuto procesu je možné použít také konvolci (viz. vztah 5.3). Příkladem takového filtru může být průměrování. V takovém případě je nová hodnota pixelu dána součtem všech v malém okolí a následně vydělena jejich počtem. Tento proces způsobí vizuálně rozmazání obrazu, což znamená potlačení vyšších prostorových frekvencí. Tak je možné potlačit šum, ovšem také dochází k často nežádoucímu potlačení ostrých přechodů, tedy hran v obrazu. Jedním z řešení je použití filtrů nelineárních.

Nelineární filtrace

U nelineárních filtrů neplatí definice zmíněná v výše. Mezi často používané filtry tohoto typu patří medián, lokace maxima, rotující maska a další [21].

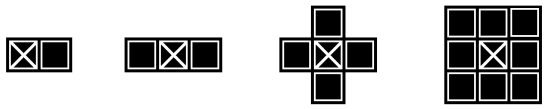
Bilaterární filtr je dalším z nelineárních filtrů. Jeho hlavní výhodou je zachování hran a zároveň potlačení vysokofrekvenčního šumu. Jsou definovány 2 parametry, které určují sílu dolní propusti. Prvním je prostorová vzdálenost, tedy jak daleko jsou od sebe dva pixely. Druhým je barevná vzdálenost, což vyjadřuje rozdíl jasových úrovní dvou pixelů. Samotná filtrace je podobná Gaussovu filtru, s tím rozdílem, že na základě prostorové a barevné vzdálenosti daný pixel okolí váhován. V případě velkého rozdílu barev se takový pixel na výsledku podílí jen minimálně, nebo dokonce vůbec. [18]

5.1.4 Matematická morfologie

Jedná se o techniku zpracovávání geometrických struktur, původně založenou na teorii množin[11]. Obraz je těmito operacemi zpracováván na základě tvarů v něm obsažených. Existuje celá řada morfologických operací, zde budou popsány základní, použité dále v této práci.

Strukturální element

V případě morfologických operací je třeba definovat strukturální element. Jedná se o množinu bodů v mřížce obrazu, která má definovaný střed. Tři příklady strukturálních elementů jsou na obrázku 5.2, vyplněná oblast znamená aktivní prvek a křížek označuje počátek.[11]



Obr. 5.2: Příklady jednoduchých strukturálních elementů.

Dilatace a eroze

Dilatace je morfologická operace provádějící přírůstek strukturálního elementu přiloženého středem na každý světlý bod v obrazu. **Eroze** je duální operací k dilataci, provádí odstranění světlých bodů, které neodpovídají přiložené masce.[11]

Otevření a uzavření

Otevření je postupné provedení eroze a dilatace s využitím totožného strukturálního elementu. Tuto operaci je možné použít k odstranění malých objektů, či rozpojení tenkých spojů. Duální operací k otevření je **uzavření**, které se provádí nejprve dilatací následovanou erozí opět se stejným strukturálním elementem. Uzavření je

možné použít k odstranění děr v objektech, nebo k propojení blízkých objektů.[11]

OpenCv: Knihovna nabízí funkci *getStructuringElement(...)* pro tvorbu strukturálního elementu tvaru obdélníku, kříže, nebo elipsy. Funkce *morphologyEx(...)* pak zajišťuje provedení zvolené morfologické operace a to na binárním, šedotónovém i barevném obraze.

5.1.5 Houghova tronsformace

K detekci jednoduchých geometrických primitiv jako jsou přímky, či kružnice, se s výhodou používá Houghova transformace. Tato metoda je založena na znalosti matematického popisu hledaného geometrického prvku o n proměnných. Vytvoří se tzv. prostor příznaků o n rozměrech podle proměnných funkce hledaného tvaru. Metoda je citlivá i na neúplné a přerušované tvary.
Přímku je možné zapsat v polárních souřadnicích (vztah 5.4). Převod do prostoru příznaků probíhá následovně:

1. Vytvoření obrazu hran. Výsledek je binární obraz.
2. Každý pixel hrany se mapuje do prostoru příznaků jako sinusoida.
3. Nalezení lokálních maxim v prostoru příznaků → nalezeny přímky.

[12]

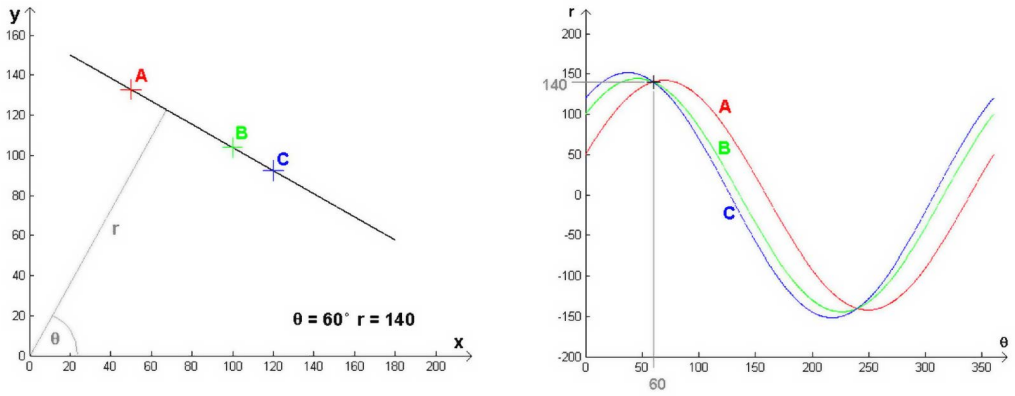
$$r = x \cdot cos\theta + y \cdot sin\theta \tag{5.4}$$

kde, r - délka normály od přímky, θ - úhel mezi osou x a normálou.[16]

OpenCv: K dispozici je implementace Houghovy transformace pro vyhledávání přímek *HoughLines* a také rozšířená verze, která vrací počáteční a koncový bod nalezené úsečky *HoughLinesP*.

5.1.6 Automatická rotace

Pro segmentaci číselníku elektroměru je třeba zajistit, aby byla čísla ve vodorovné pozici. Tento požadavek může být zajištěn více způsoby. Nejjednodušší je pořídit snímek vždy správně otočen, to ovšem klade nároky na uživatele. Další možností je využít senzoru natočení, který se dnes v mobilních zařízeních běžně vyskytuje. Třetí možností je provést korekci rotace až na pořízeném snímku. Čelní panel elektroměru obsahuje několik řádků textů a také více vodorovných čar s jejichž využitím

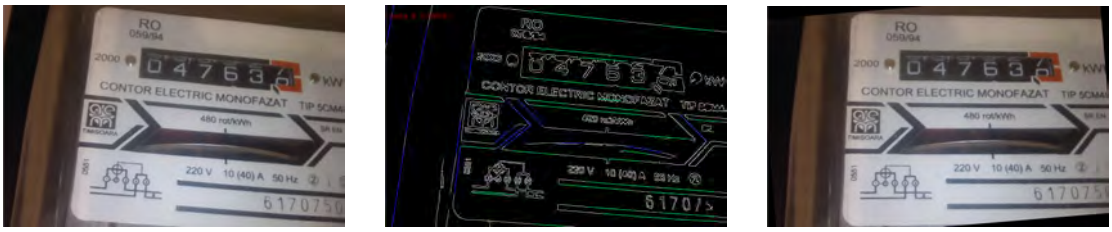


(a) (b)

Obr. 5.3: Příklad Houghovy transformace. (a) Úsečka v prostoru obrazu (x, y), (b) prostor příznaků (θ, r). [16]

je možné korekci rotace provést. V projektu elektroměru byla implementována metoda automatického otočení s využitím Houghovy transformace (viz. kapitola 5.1.5). Postup:

1. Převod barevného obrazu na šedotónový.
2. Detekce hran (Canny).
3. Nalezení nepřerušených čar delších než definované minimum.
4. Lokalizace dominantních čar → průměrný úhel otočení.
5. Rotace obrazu elektroměru.



(a) (b) (c)

Obr. 5.4: Automatická korekce rotace. a) Originální obraz, b) nalezené čáry v obrazu hran, c) otočený obraz na základě natočení dominantních čar.

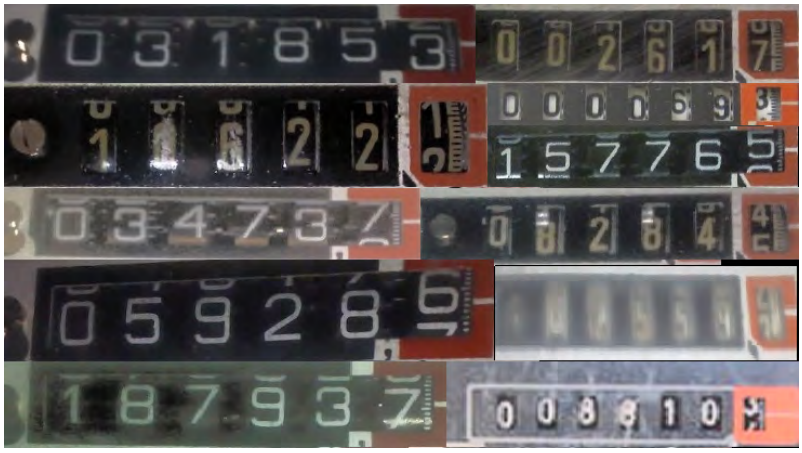
Implementace: Automatickou rotaci implementuje metoda třídy *mycv* s názvem *mycvRotateAcDominantLines(...)*. Vstupem je šedotónový obraz a vrací rotační matici a úhel rotace.

5.2 Segmentace

Segmentací se obecně rozumí oddělení objektů v obrazu od pozadí, či samých od sebe. Ideálně jsou výsledkem nepřekrývající se oblasti ohraničující jednotlivé objekty nacházející se na obrazu. V případě této práce je třeba stanovit umístění počítadla, či počítadel, elektroměru v obrazu. Následně rozdělit snímek počítadla na jednotlivé číslice. V případě segmentace číselníku byly vypracovány dvě metody, první využívá apriorních znalostí o hledaném objektu a druhá kaskádový klasifikátor.

5.2.1 Segmentace číselníku verze 1

Hledaný objekt, tedy počítadlo elektroměru, může mít u různých typů elektroměrů jisté odlišnosti, ovšem řadu znaků mají takřka všechny společnou. Na obrázku 5.5 je ukázka několika výřezů počítadel běžných elektroměrů. Většina se vyznačuje černým pozadím s bílým textem a červeným rámečkem u desetinného místa. Existují ovšem typy, které mají pozadí číselníku bílé a tmavé číslice, nebo takové, kde desetinné místo není barevně odděleno. Algoritmus lokalizace číselníku by v ideálním případě reagoval na všechny typy.



Obr. 5.5: Počítadla běžných elektroměrů.

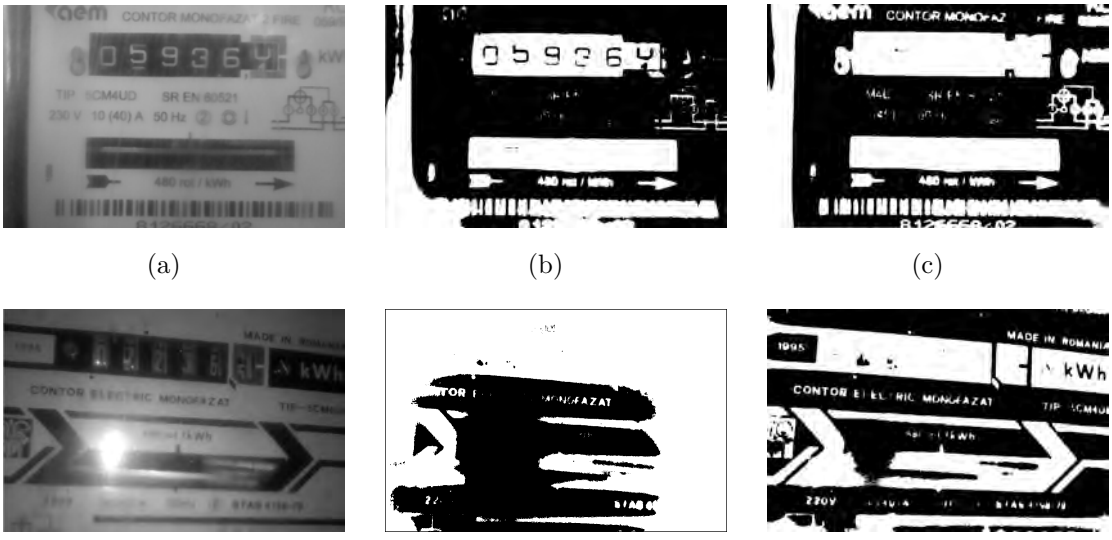
Postup lokalizace počítadla na snímku elektroměru bude následující:

1. Korekce rotace (viz. kapitola 5.1.6).
2. Tvorba masky pozadí číselníku.
3. Detekce vertikálních čar.
4. Nalezení shluků čar - texty v obrazu.
5. Odstranění shluků, neodpovídajících číselníku.

Jakmile je provedena korekce rotace (viz. kapitola 5.1.6), texty jsou tedy ve vodorovné pozici, jsou na základě znalosti pozadí číselníku elektroměru vytvořeny binární masky, které budou použity k upřesnění polohy číselníku. Jedna z červené složky obrazu a druhá zahrnuje velmi tmavé oblasti.

Prahování

Prahování znamená snížení jasových úrovní v obraze, často se používá binární prahování, pak výstupní obraz má jen dvě hodnoty jasu (černá/bílá). Pro účely této práce je důležité právě binární prahování. Princip převodu na černobílý obraz spočívá ve stanovení mezní hodnoty jasu, tedy prahu. Pixely s jasovou hodnotou vyšší než prah jsou označeny jako bílé a ostatní jako černé. Stanovení hodnoty prahu je tedy klíčové, možné je využít maxima a minima jasové hodnoty, nebo tvaru histogramu. Na obrázcích 5.6 jsou dva příklady prahování. První obraz (a) je prahovaný konstantním prahem, výsledek (b) uspokojivě oddělil tmavé a světné části obrazu. Druhý příklad (d) demonstruje problém s nerovnoměrným osvětlením. Zde prahování s konstantním prahem selhává (e). Řešením tohoto problému je adaptivní prahování. Hodnota prahu každého pixelu je určena lokální statistikou okolí. Často se používá střední hodnota, medián a další. Výsledky adaptivního prahování zmíněného příkladu pro okolí velikosti čtvrtiny šířky obrazu jsou na obrazech 5.6 (c) a (f). V druhém případě je patrné, že adaptivní prahování velmi dobře potlačuje nerovnoměrné osvětlení scény.



(a) (b) (c)

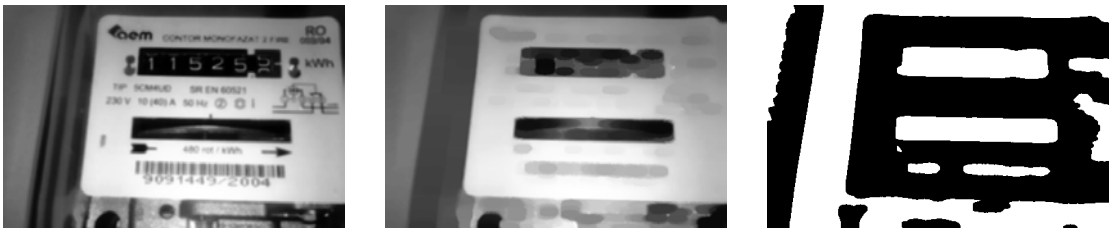
(d) (e) (f)

Obr. 5.6: Příklad prahování dvou obrazů. (a, d) Originální obrazy ve stupních šedě, (b, e) jednoduché prahování, (c, f) adaptivní prahování.

OpenCv: V knihovně se nachází implementace jak základního *threshold(...)*, tak adaptivního *adaptiveThreshold(...)* prahování.

Tvorba masky pozadí číselníku

Cílem je vytvořit binární masku, která odstraní oblasti neobsahující počítadlo. Pozadí číselníku běžných elektroměrů je černé (obrázek 5.5), takže masku je možné vytvořit s využitím adaptivního prahování (kapitola Prahování). Šedotónový obraz elektroměru je nutné předzpracovat. Prvním krokem je odstranění šumu, což zajistí bilaterární filtr, který zachovává ostré přechody (kapitola 5.1.3). Další je odstranění detailů v obraze, což potlačí texty a maska tedy bude představovat jen pozadí. Tuto operaci zajistí morfologické uzavření (kapitola 5.1.4) se strukturálním elementem tvaru elipsy s dvojnásobnou horizontální osou oproti vertikální (inspirováno [4] Chapter 5.). Takto upravený obraz je připraven k prahování (obrázek 5.7 (b)). Binární maska snímku je na obrázku 5.7 (c).



(a) (b) (c)

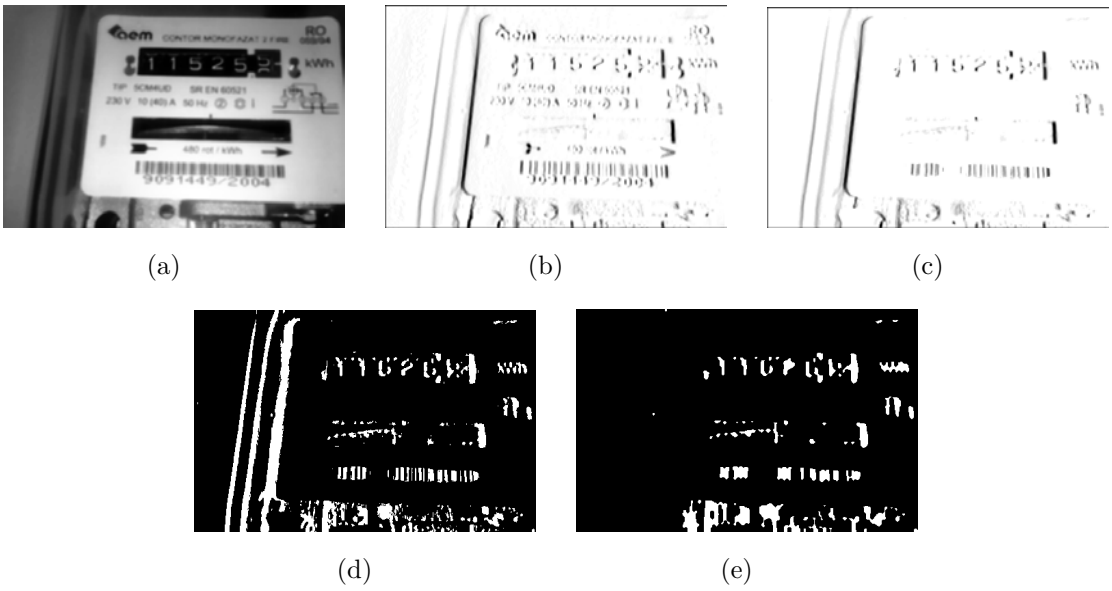
Obr. 5.7: Úprava obrazu pro tvorbu masky pozadí. (a) šedotónový obraz elektroměru, (b) odstraněné detaily morfologickým uzavřením, (c) výsledek prahování.

Detekce vertikálních hran

V kapitole 5.1.3 je uveden vzorec 5.3 pro 2D diskrétní konvoluci, která kromě filtrace šumu může být použita i k dalším účelům. V tomto případě je třeba v obrazu lokalizovat vertikální hrany. K tomuto účelu se hodí některý z konvolučních hranových detektorů, založených na aproximaci derivace obrazu, jako jsou Prewitt, Sobel(rovnice 5.5), Roberts, Laplace a další [21]. Tyto detektory je možné orientovat na určitý směr a tak získat aproximaci derivace pouze v určitém směru, obvykle vertikálním, či horizontálním.

$$h_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, h_2 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}, h_3 = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \dots \tag{5.5}$$

Pro účely tohoto projektu byl použit operátor Sobel. Odezva operátoru je vidět na obrázku 5.8 (b). Kromě ostrých hran vertikálních čar textů, symbolů a okrajů elektroměru se projevil na výstupu hranového detektoru i šum, který v obraze zbyl po filtraci. Tento šum má menší úroveň, než skutečné hrany, pro jeho potlačení postačí odstranit všechny hodnoty nedosahující určité úrovně. Tomuto procesu se říká částečné prahování, výstupem není binární obraz, jako tomu bylo u prahování v kapitole 5.2.1. Vhodná hodnota pro odstranění šumu v obrazu hran byla experimentálně stanovena na dvacetinu rozsahu hodnot. Po odstranění nízkých úrovní je použita binární maska vytvořená v předchozím kroku, výsledek těchto operací je na obrázku 5.8 (c). Takto připravený obraz je vhodný pro prahování.



(a) (b) (c)

(d) (e)

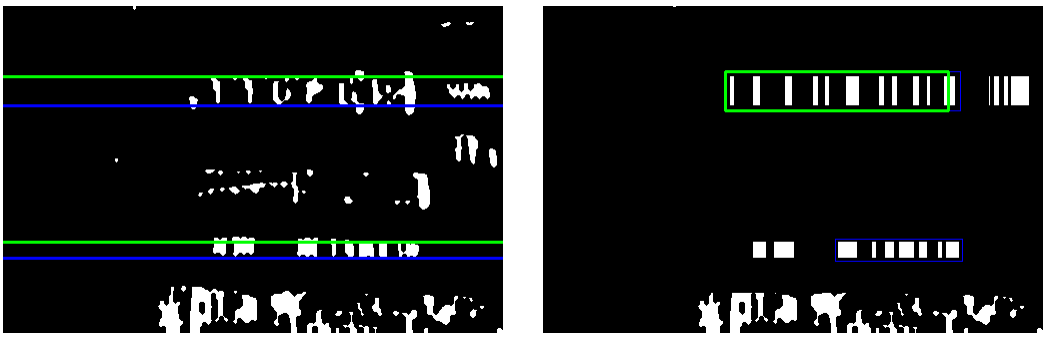
Obr. 5.8: Obraz vertikálních hran použitím Sobel operátoru. (a) šedotónový obraz elektroměru, (b) obraz vertikálních hran, (c) aplikovaná binární maska a odstranění nízké hladiny šumu, (d) prahovaný obraz hran, (e) odstraněny příliž velké objekty.

OpenCv: Knihovna obsahuje funkci *Sobel(...)*, která implementuje hranový detektor typu Sobel. Pro použití dalších typů operátorů je možné použít funkci *filter2d(...)*, jenž provede konvoluci vstupního obrazu s definovanou maskou.

Další krok je prahování předzpracovaného hranového obrazu. Klasické prahování zde selhalo, protože silnější hrany jako okraje elektroměru a větších obrazců někdy potlačily hledané hrany číslic. Z tohoto důvodu je použito adaptivní prahování (obrázek 5.8 (d)).

Lokalizace

Problém určení polohy číselníků elektroměru v této fázi znamená nalezení série vertikálních čar, které budou mít dostatečný počet, vzájemnou vzdálenost a jejichž rozměr vyhovuje poměru stran číselníku. Nejprve jsou označeny potenciální oblasti řádků textu na základě počtu objektů v daném řádku. Tyto oblasti jsou označeny na obrázku 5.9 (a), kde zelená čára označuje počátek a modrá konec. Následně jsou procházeny jednotlivé nalezené řádky a v nich jsou vyplňovány sloupce obsahující větší než stanovený počet světlých pixelů, výsledek je vykreslen bíle na obrázku 5.9 (b). Posledním krokem je nalezení horizontálních hranic textu, respektive číselníku, to je provedeno propojením jednotlivých vertikálních čar na základě jejich vzdálenosti. Postupuje se v každém řádku zleva doprava, pokud je nalezena vertikální čára, označí se její počátek. Je-li v definované vzdálenosti vpravo další vertikální čára, ponechá se počátek a označí konec. Tímto způsobem se pokračuje, dokud jsou nacházeny další blízké čáry, nebo na konec obrazu. Výsledkem jsou označené oblasti možných textů (obrázek 5.9 (b) modrá).



(a) (b)

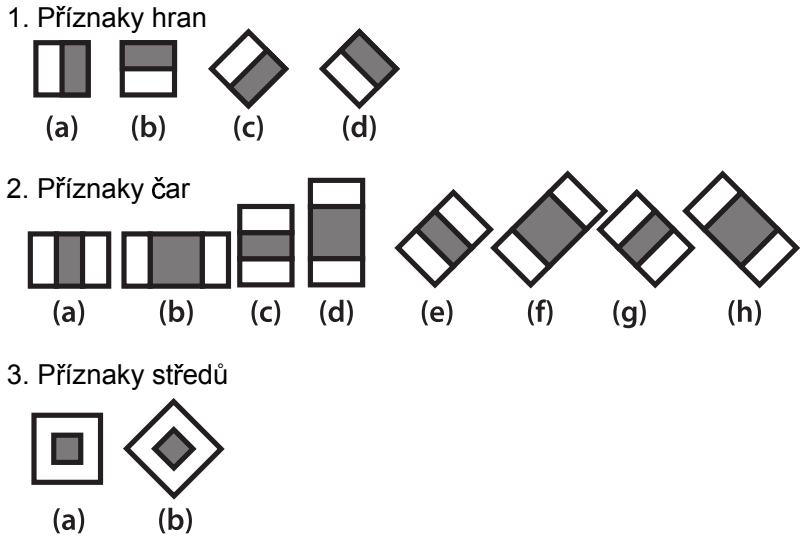
Obr. 5.9: Demonstrace lokalizace číselníku z obrazu hran. (a) Vyznačené oblasti řádků potenciálních textů, (b) ohraničení možných číselníků (modře) a oblast číselníku (zelená).

5.2.2 Segmentace číselníku verze 2

V předchozí kapitole 5.2.1 je popsána metoda lokalizace číselníku, která pracuje s apriorními znalostmi a dala by se označit za deterministickou. Druhá verze využívá aparát strojového učení, v tomto případě se jedná o učení s učitelem. Konkrétně se jedná o kaskádový klasifikátor, který určí, zda se v předloženém obraze nachází, či nenachází hledaný objekt. Samotná lokalizace je prováděna postupným skenováním obrazu pomocí okna o definovaném rozsahu velikostí a natočení.

Příznaky typu Haar

Na obrázku 5.10 jsou zobrazeny příklady masek příznaků typu Haar. Pomocí těchto masek je možné převést vstupní obraz na prostor příznaků. Výpočet hodnoty jednoho pixelu je následující, nejprve je vypočten rozdíl sum jasových hodnot obrazu dvou oblastí odpovídajících (šedá (odečíst) a bílá (přičíst) na obrázku 5.10) přiložené masce. Po aplikaci prahu na vypočtený rozdíl je výsledná hodnota dvoustavová (0/1, nebo -1/1). Aplikací jednoho takového příznaku na celý obraz tedy vznikne obraz binární. Tento výsledek se dá použít jako velmi jednoduchý klasifikátor, např. při použití masky příznaků hran (a) (obr. 5.10) bude výsledek lokalizovat vertikální hrany. Výpočet je navíc velmi rychlý.



Obr. 5.10: Příklady masek příznaků typu Haar.[5]

AdaBoost

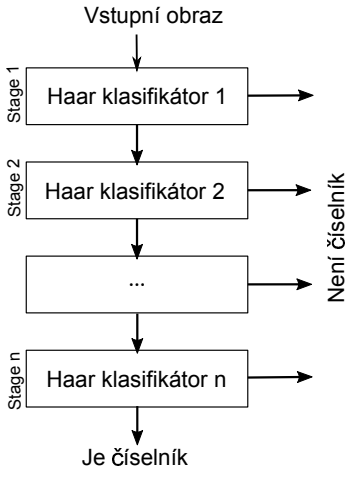
Metoda AdaBoost (Adaptive Boosting) je jednou z metod, která provádí efektivní učení klasifikátorů a zároveň selekci a váhování příznaků. Pracuje s klasifikátory typu weak learner (špatný žák), který má co se týče klasifikace úspěšnost jen o něco lepší než náhodné rozhodování. Cílem je spojit více weak learner klasifikátorů do jednoho dobrého (strong learner). Pro detekci číselníku byly jako weak learner klasifikátory použity příznaky typu Haar.

Je třeba zavést dva pojmy, prvním je „Hit rate“ (dále HR) značící úspěšnost správného přiřazení a druhým uvFalse alarm (dále FA), který vyjadřuje míru mylné klasifikace. V případě Haar příznaků jsou hledány takové, které mají HR blízký 100 %, ovšem FA se obvykle blíží 50 %. To znamená, že jsou sice označeny správně hledané oblasti, ale společně s nimi je označeno také mnoho oblastí za hledané,

i když jimi nejsou. Když se spojí takových klasifikátorů více do série (např. 20), přesnost HR se zmenší jen nepatrně ($0,999^{20} \approx 98\%$), ale FA se rapidně snižuje ($0,5^{20} \approx 0,0001$).[5]

Kaskádový klasifikátor

V roce 2001 byla publikována metoda rychlé lokalizace objektů v obraze založena na boosting algoritmech pracujícími s jednoduchými příznaky (Haar) [25]. Tato metoda poskytuje robustní detekci objektů v reálném čase. Jedná se o algoritmus učení s učitelem, jsou tedy vyžadována trénovací data. Prvním typem trénovacích dat jsou obrazy obsahující hledaný/é objekty s jejich ohraničením. Těchto vzorů musí být dostatečný počet (stovky až tisíce) a měly by zahrnovat širokou škálu možností hledaného objektu (různé osvětlení, pozadí, typy, ...). Druhou množinou vstupních dat jsou obrazy, kde se hledaný objekt nenachází. To zahrnuje různé druhy pozadí, kde by se mohl hledaný objekt nacházet.
Struktura kaskádového klasifikátoru je znázorněna na obrázku 5.11. Vstupní obraz je postupně zpracováván jednotlivými klasifikátory od stupně 1 výše až do konečného. Nejnižší stupeň obsahuje nejjednodušší typ klasifikátoru, buď jednoduchý rozhodovací strom, nebo jediný Haar klasifikátor. Pokud vstupní obraz neprojde prvním stupněm, je vyřazen, jinak pokračuje dál. Pokud projde až na konec, tak je tento obraz klasifikován jako hledaný objekt (číselník). Rychlost tohoto přístupu je zajištěna efektivním „zahazováním“ obrazů bez hledaného objektu, takové se nedostanou obvykle přes prvních několik stupňů.



Obr. 5.11: Blokobé schéma struktury kaskádového klasifikátoru.

OpenCV: Knihovna poskytuje třídu *CascadeClassifier*, která umožňuje načtení vytvořeného kaskádového klasifikátoru a pomocí něj nalézt hledaný objekt/ty v obraze. Metoda *detectMultiScale(...)* této třídy zajišťuje efektivní skenování vstupního

obrazu, je možné nastavit rozsah velikostí hledaného objektu. K vytvoření klasifikátoru slouží konzolové aplikace *objectmarker*, *createsamples*, *haartraining*. Pomocí nich je možné vytvořit tréningová data a provést trénování klasifikátoru.

Vytvoření klasifikátoru

Tvorba kaskádového klasifikátoru je výpočetně náročná úloha. Aby bylo trénování úspěšné, je třeba zajistit dostatečný počet dat. Jedná se o obrazy hledaného objektu a také pozadí, na kterém by se mohl tento objekt nacházet. Doporučují se tisíce vzorů a dvojnásobek snímků pozadí [6]. V případě vyhledávání číselníku elektroměru bylo použito 200 vzorů číselníků a 400 obrázků pozadí, protože nebyl dostupný větší počet dat. Sada snímků pozadí, byla vytvořena odstraněním číselníku ze snímků analogových elektroměrů a byla doplněna sadou elektroměrů digitálních, které rovněž nemají být detekovány.

5.2.3 „Real time“ detekce číselníku

Výše popsané metody budou použity k lokalizaci v jednotlivých snímcích kamery zařízení. Detekce by tedy měla být dostatečně rychlá, aby byla video-sekvence plynulá. Zajistit tento požadavek je možné zmenšením obrazu. Tak bude provedená „hrubá“ lokalizace, jakmile bude pořízen snímek, bude pozice upřesněna na základě tmavých okrajů a případně červeně ohraničeného desetinného místa.
Při prvních testech aplikace byl pozorován jev „blikání“ nalezeného rámečku číselníku, také se chvílemi skokově měnila jeho poloha stále v těsné blízkosti hledaného číselníku. K potlačení tohoto jevu byl implementován algoritmus průměrování polohy číselníku přes sekvenci snímků. Zpracování probíhá na každém snímku videa kamery. Nejprve je provedena detekce číselníku na aktuálním snímku, následuje postup průměrování nalezených obdélníků. Je vytvořen seznam obsahující položky s parametry: obdélník, délka života, délka prodlevy. Pokud je seznam prázdný, jsou nalezené obdélníky vloženy. Není-li prázdný provede se:

- Je-li vstupní obdélník v překryvu alespoň 60 % s některým v seznamu aktualizují se hodnoty na průměr velikosti a polohy obou obdélníků. Také je přičtena hodnota k délce života.
- Není-li vstupní obdélník v překryvu, je vložen do seznamu.
- Pro každý obdélník v seznamu, jenž nebyl aktualizován, přičti prodlevu. Překročí-li prodleva definovanou hodnotu (15) je položka odstraněna.

Pokud některý z obdélníku překročí definovanou hodnotu délky života (10) je vyhodnocen jako číselník a proces rozpoznání je spuštěn.

Implementace: Výše popsaný algoritmuse je vytvořen přímo v Java třídě provádějící snímání kamerou *CamActivity*, strukturu položek seznamu obdélníků tvoří třída *FrameParams*.

5.2.4 Úspěšnost lokalizace číselníku

Aby bylo možné stanovit úspěšnost lokalizace, bylo z celkového počtu snímků odděleno 40 snímků. Testování bylo v první fázi provedeno pouze na jednotlivých statických obrazech prostřednictvím testovacího software. V tomto případě byly výsledky první metody velmi dobré (92.5 %), ovšem v případě metody druhé byla úspěšnost nalezení číselníku menší než 50 %. Slabý výsledek druhé metody je patrně způsoben malým počtem tréningových dat, doporučují se tisíce vzorů [6] a v případě této práce bylo k dispozici jen 200 vzorů. Kdyby bylo zajištěno více vzorů (alespoň 1000), úspěšnost by vzrostla.
Po implementaci algoritmů lokalizace číselníku do mobilního zařízení, byly provedeny testy v „real time“ režimu . Zde jsou výsledky, zejména u druhé metody, výrazně lepší. Díky algoritmu upřesňování polohy v sekvenci snímků (kapitola 5.2.3) je detekce číselníku blízká úspěšnější.

	Úspěšnost detekce číselníku [%]	
Metoda	Statický obraz	Dynamický obraz
Thresh&Verticals	92,5	97,5
HaarCascade	47,5	90

Tab. 5.1: Výsledky testů detekce číselníku elektroměru.

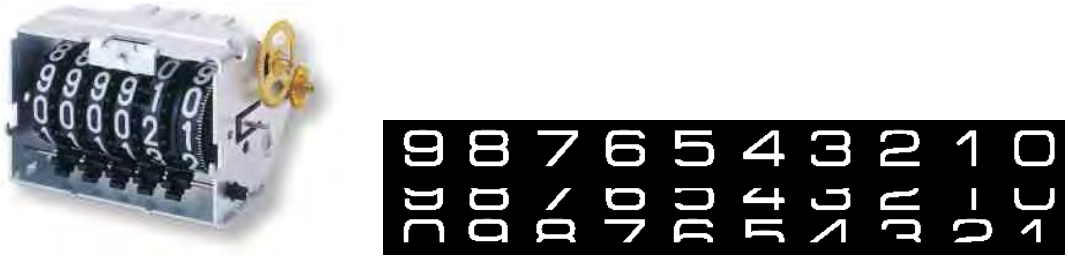
Implementace: První metoda detekce číselníku je implementována v C++ ve třídě *mycv* metodou *detectFrame(...)*. Druhá metoda je v Java kódu ve třídě *CamActivity*, protože je tak jednodušší přístup k souboru obsahujícím naučený klasifikátor.

5.2.5 Segmentace číslic

V předchozí kapitole byl popsán postup získání výřezu počítadla. Následuje oddělení jednotlivých číslic, které budou dále rozpoznávány. Separace číslic může být provedena více způsoby (články [17], [26], [4]). Všechny metody pracují s výřezem počítadla elektroměru, jehož postup tvorby je uveden v předchozí kapitole 5.2.1. Pro segmentaci jednotlivých číslic je důležité předzpracování, které potlačí šum a zvýrazní a zaostří jednotlivé číslice.

Specifikace číselníku

Na obrázku 5.5 je několik ukázek číselníků elektroměrů několika běžných typů. Jedná se o rotační mechanické zobrazovače. Na povrchu válce rotujícího v horizontální ose čela elektroměru jsou rozmístěny číslice od 0 do 9 (obrázek 5.12 (a)). Obdélníkovým průhledem v čele elektroměru není vždy vidět jen jedna číslice, ale někdy části dvou sousedních. Při rozpoznávání se s tím musí počítat, takže vzory pro porovnání mohou vypadat jako na obrázku 5.12 (b). V prvním řádku jsou číslice od 0 do 9 a na druhém řádku jsou vytvořeny symboly vzniklé spojením přibližně 2/3 dvou po sobě jdoucích čísel. Je zřejmé, že v takovém případě bude možné odečítat hodnotu s přesností 0.5 zobrazované hodnoty.



(a) (b)

Obr. 5.12: (a) Vnitřní mechanizmus počítadla elektroměru [15], (b) vzor číslic elektroměru.

Předzpracování

Na obrázku 5.5 je znázorněna řada počítadel elektroměrů. Tyto snímky jsou zatíženy nejen aditivním šumem, ale také zkreslením v důsledku nerovnoměrného, či nedostatečného osvětlení a v několika případech také odlesky způsobené LED přisvícením fotoaparátu. V předzpracování je ideálně třeba provést prahování snímku tak, aby výsledkem byl binární obraz obsahující pouze jednotlivé číslice počítadla.

Prahování - byly navrženy dvě metody, které potlačují rušivé vlivy, zejména nerovnoměrné osvětlení a aditivní šum, zároveň zachovávají kontury číslic. Snímek nejprve převeden do stupňů šedi.

V první metodě je postup následující:

1. Ostření: Od šedotónového obrazu je odečten obraz upravený filtrem gaussova typu (kapitola 5.1.3), tato operace zvýrazní kontury objektů (obr. 5.13 (b, f)).
2. Filtrace aditivního šumu: Lineární filtry by sice potlačily šum, ale také hrany, které chceme zachovat, proto byl použit nelineární bilaterární filtr (kapitola 5.1.3), který zachovává hrany (obr. 5.13 (c, g)).
3. Adaptivní prahování: Po odstranění šumu je provedeno prahování (obr. 5.13 (d, h)).

(a) (b) (c) (d)

(e) (f) (g) (h)

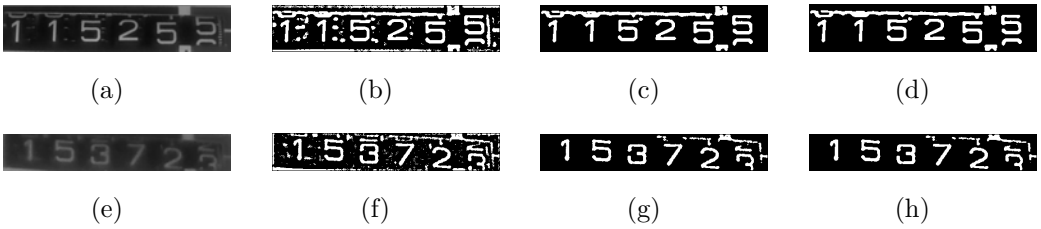
Obr. 5.13: Příklad prahování číselníku metoda 1. (a, e) Šedotónový obraz, (b, f) zaostřený obraz, (c, g) filtrace šumu, (d, h) adaptivní prahování.

Druhá metoda:

1. Adaptivní prahování: Oproti předchozí metodě je prahován přímo šedotónový obraz bez filtrace.
2. Odstranění objektů příliš malých rozměrů: jak je vidět z prahovaného obrazu (obr. 5.14 (b, f)), nachází se zde velké množství malých objektů způsobených šumem, odlesky a někdy defekty na elektroměru. Optimální úrovně velikostí byly stanoveny experimentálně na 1/9 počtu řádků pro osu x a 1/8 také počtu řádků pro osu y, tedy vše co má menší rozměry je odstraněno a výsledek je na obrázku (g, h).
3. Mediánová filtrace: Odstranění šumu je provedeno mediánovým filtrem v binárním obraze, výsledeky obr. 5.14 (d, h).

Pozn.: Pokud se prohodí kroky 2 a 3, metoda je rychlejší, protože po mediánové filtraci se v obraze nachází výrazně méně objektů a zjišťování velikosti je rychlejší. Tento postup ovšem v některých případech způsobil deformaci číslic, když se v jejich blízkém okolí nacházela malá porucha (odlesk, poškození panelu, ...).

Další zpracování je společné pro obě popsané metody. Je třeba odstranit pruhy ohraničující číslice, ovšem aby se zachovala čísla je třeba nejprve oddělit je od těchto okrajů. K tomu účelu je použitá horizontální projekce (obr. 5.15 (b, f)). Jedná se o graf sum světlých pixelů řádků obrazu (a, e). Pro nalezení hranic se kontrolují okraje až do 1/5 výšky řádku ze shora i zespoda.



(a) (b) (c) (d)

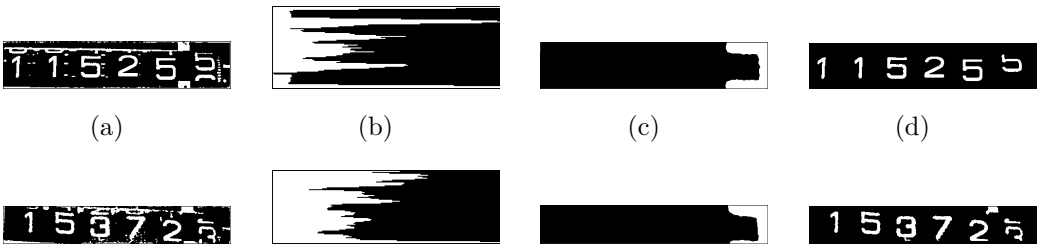
(e) (f) (g) (h)

Obr. 5.14: Příklad prahování číselníku metoda 2. (a, e) Šedotónový obraz, (b, f) adaptivní prahování, (c, g) odstranění malých objektů, (d, h) mediánový filtr.

Postup nalezení hranic je následující:

1. Nalezení maxima
2. Je-li větší než 3/4 šířky obrazu: Vyhledat nejbližší minimum směrem ke středu. Jinak pro aktuální stranu ukončit.
3. Odstranit objekty mezi nalezeným minimem a okrajem obrazu.

Po odstranění ohraničení je ještě vymazán červený rámeček u desetinného místa. K tomuto účelu je vytvořena maska obsahující pouze červené oblasti (obr. 5.15 (c, g)). Výsledek těchto operací je na obr. 5.15 (d, h). Tím je prahování a předzpracování hotové, následuje samotná segmentace.



(a) (b) (c) (d)

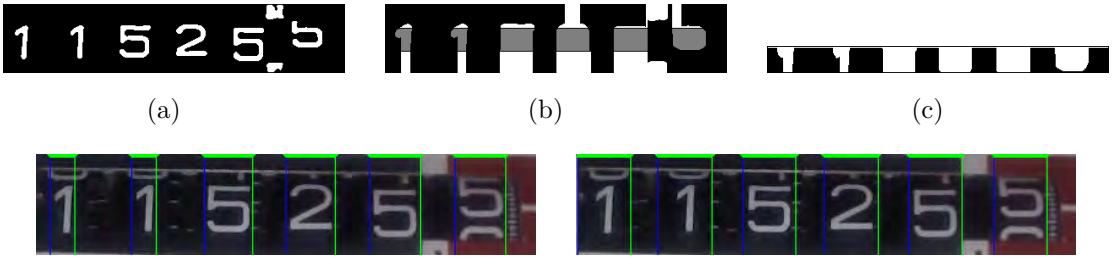
(e) (f) (g) (h)

Obr. 5.15: Demonstrace odstranění okrajů a červeného rámečku. (a, e) Prahovaný obraz, (b, f) horizontální projekce (zvetšeno), (c, g) červená maska, (d, h) výsledek.

Vertikální projekce

V článku [26] je popsán způsob segmentace znaků ze státních poznávacích značek. Tento problém je podobný oddělování číslic elektroměru, proto byla tato metoda použita i zde. Po provedení prahování obrazu (obr. 5.16 (a)) číselníku elektroměru jsou vyplněny, s využitím morfologie a kontur, malé díry v objektech. Následně je z obrazu vyříznuta střední třetina v horizontálním směru (b), to zajistí separaci znaků od případných zkreslení na horním, či dolním okraji. Z takto upraveného obrazu

je vytvořen graf, vyjadřující sumy světlých pixelů jednotlivých sloupců (vertikální projekce) (c). Následuje separace oblastí z tohoto grafu. To je dosaženo nalezením úseků s minimální hodnotou, optimální úroveň, brána za minimum, byla stanovena na pětinu rozsahu. Nyní je separace ukončena, dva sousední minimální úseky vždy označují hledaný objekt (d). V některých případech může předchozí zpracování způsobit špatnou šířku oblasti, proto je provedena normalizace šířky na základě střední hodnoty (e).



(a) (b) (c)

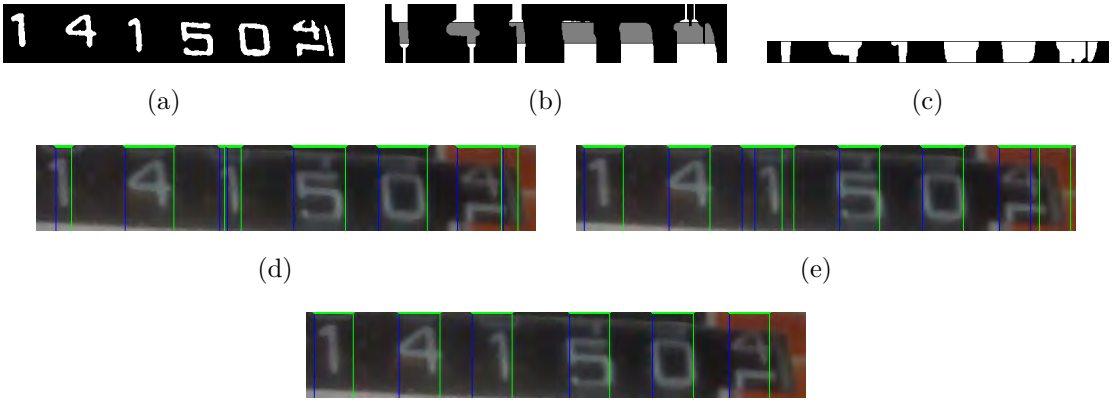
(d) (e)

Obr. 5.16: Segmentace číslic - vertikální projekce. (a) Binární předzpracovaný obraz, (b) vyplněné oblasti - šedě označena střední třetina, (c) vertikální projekce, (d) nalezené číslice, (e) normalizace šířky oblastí.

Na obrázku 5.16 je uveden příklad segmentace, kde byly v předzpracování odstraněny všechny nežádoucí objekty. Bohužel tomu tak nebude vždy. V takovém případě se budou mezi výslednými separovanými oblastmi číslic nacházet i jiné oblasti, které je třeba odstranit. Normalizace šířky způsobí překryvy některých oblastí. Vyhledání překryvů je zajištěno postupným porovnáváním souřadnic úseků navzájem. Je-li nalezen překryv dvou oblastí, je ta vpravo odstraněna. Situace je znázorněna na obr. 5.17.

Vyhledávání kontur

Kontury označují hranice jednotlivých objektů v obrazu, jejich vhodnou selekcí je možné oddělit požadované objekty od pozadí a šumu. Vstupem je, obdobně jako u předchozí metody, binární obraz výřezu číselníku (obr. 5.15). Pomocí morfologických operací jsou propojeny blízké objekty nad sebou, což odpovídá symbolu složenému ze dvou částí pootočených číslic (druhý řádek 5.12). Nejprve je provedeno morfologické uzavření s použitím úzkého vysokého strukturálního elementu, což zajistí spojení případně oddělených částí znaku. Dále je provedena dilatace malým strukturálním elementem tvaru elipsy. Na takto připraveném obrazu jsou vyhledány všechny externí kontury, to znamená, že nejsou brány v úvahu otvory a vnitřní objekty. Pro každou konturu je nalezen opsaný obdélník a dále se bude pracovat jen s nimi.



(a) (b) (c)

(d) (e)

(f)

Obr. 5.17: Segmentace číslic - vertikální projekce. (a) Binární předzpracovaný obraz, (b) vyplněné oblasti - šedě označena střední třetina, (c) vertikální projekce, (d) nalezené číslice, (e) normalizace šířky oblastí, (f) odstraněny překrývající se oblasti.

Jednotlivé kontury jsou podrobeny řadě kontrol, aby byli vyřazeny všechny neobsahující číslici. Postup kontroly:

1. Kontrola rozměrů: Jsou odstraněny příliš úzké oblasti, stejně jako velmi malé, kde by se číslice nemohla vejít.
2. Umístění středu: Pokud střed objektu leží v 1/4 vzdálenosti od horního, či dolního okraje, je vyřazen jako artefakty okraje číselníku.
3. Odstranění příliš blízkých kontur: Pokud jsou dvě kontury velmi blízko sebe, je menší z nich odebrána.
4. Normalizace: Ze zbývajících kontur je nalezeno maximum a minimum ve vertikálním směru a průměrná šířka. Všechny ohraničení objektů jsou nastaveny na tyto rozměry.
5. Oprava překryvů: Pokud se v této fázi nachází překrývající se objekty, jsou odstraněny ty, které se v dvojici překrývajících nacházejí blíže vertikálnímu okraji.

Tímto postupem bylo dosaženo uspokojivých výsledků segmentace jednotlivých znaků. Může se stát, že je za znak označena oblast, která jej neobsahuje. Obvykle to způsobují širší okraje. Pak bude tento symbol zamítnut v dalším kroku zpracování, tedy klasifikaci číslic.

Další možnosti segmentace

Probléme segmentace znaků je velmi rozšířen a bylo popsáno velké množství řešení, ať už robustních, nebo určených pro konkrétní aplikaci. Dvě v práci použité metody, zmíněné výše, jsou jedny z nejrozšířenějších a splňují účel.

5.2.6 Přesnost segmentace znaků

Obdobně jako u lokalizace číselníku i zde bylo použito čtyřiceti testovacích vzorů elektroměrů k určení přesnosti segmentace. Na každém vzoru se nachází 6 číslic, takže testováno bylo celkem 240 znaků. Výsledky jsou uvedeny v tabulce 5.2, úspěšnost byla definována vztahem 5.6.

$$\text{Úspěšnost} = \frac{Správně}{Celkem + Chybně} \cdot 100[\%] \tag{5.6}$$

		Segmentace		
Metoda	Celkem znaků	Správě	Chybně	Úspěšnost segmentace [%]
Vertikální projekce	240	236	16	92,19
Kontury	240	235	4	96,31

Tab. 5.2: Výsledky testů detekce číselníku elektroměru.

Implementace: Obě metody segmentace včetně předzpracování se nacházejí ve třídě *mycvDigitsSegm*.

5.3 OCR

V předchozí kapitole je popsán postup segmentace nejprve číselníku a pak jednotlivých číslic počítadla elektroměru. Dalším krokem je rozpoznání číslic.

5.3.1 Geometrické deskriptory

Mezi tyto deskriptory se řadí všechny geometrické vlastnosti objektů. Pro účel rozpoznání znaků je možné využít kombinace více příznaků. K vyhodnocení výsledné číslice je výhodné použít rozhodovací strom, který na základě počtu objektů a otvorů v nich rozčlení znaky do menších kategorií. Přesné určení znaku je provedeno pomocí srovnání tvaru vzoru a testované číslice na základě momentových invariantů.

Geometrické momenty

Geometrické momenty jsou jakýmsi hrubým popisem tvaru objektu na základě jeho kontury. Jde o kvantifikaci tvaru a to často nezávislou na konkrétních geometrických deformacích. Takové momenty mají invariantní vlastnosti. Další důležitou funkcí je rozlišitelnost, tedy pro různé tvary musí být výrazně jiná hodnota. Hodnota jednoho momentu obvykle neposkytuje zmíněné vlastnosti, proto se jich používá více současně. V takovém případě se jedná o vektor invariantů velikosti n. Každý objekt je pak popsán bodem v n-dimenzionálním prostoru. Základní moment je dán dvěma

parametry (p, q), které určují jeho řád. Výpočet je sumou všech pixelů kontury objektu dle vztahu 5.7. [5]

$$m_{p,q} = \sum_{i=1}^{n} I(x,y) x^p y^q \tag{5.7}$$

kde $m_{p,q}$ - moment řádu (p, q), p - řád ve směru řádků, q - řád ve směru sloupců, x, y - souřadnice bodů kontury v obraze, $I(x,y)$ - jasové úrovně, n - počet bodů kontury.[5]

Takto vypočtené momenty jsou ovšem závislé na souřadnicovém systému, takže není zajištěna invariance např. vůči rotaci. Z toho důvodu se používají normalizované momenty. Pro jejich výpočet je nejprve třeba definovat centrální momenty dle vztahu 5.8.

$$\mu_{p,q} = \sum_{i=1}^{n} I(x,y)(x - x_{avg})^p (y - y_{avg})^q \tag{5.8}$$

kde $x_{avg} = m_{10}/m_{00}$ a $x_{avg} = m_{01}/m_{00}$. Vydělením příslušnou mocninou momentu m_{00} je možné získat normalizovaný moment (vztah 5.9).

$$\eta_{p,q} = \frac{\mu_{p,q}}{m_{00}^{(p+q)/2+1}} \tag{5.9}$$

Geometrické momentové invarianty jsou definovány lineárními kombinacemi normalizovaných momentů. Vztahy 5.10 představují 7 momentových invariantů, podle tvůrce [14] nazvaných Hu momenty.

$$\begin{aligned}
h_1 &= \eta_{20} + \eta_{02} \\
h_2 &= (\eta_{20} - \eta_{02})^2 + 4 \cdot \eta_{11}^2 \\
h_3 &= (\eta_{30} - 3 \cdot \eta_{12})^2 + (3 \cdot \eta_{21} - \eta_{03})^2 \\
h_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
h_5 &= (\eta_{30} - 3 \cdot \eta_{12})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3 \cdot (\eta_{21} + \eta_{03})^2) + \\
&\quad + (3 \cdot \eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})(3 \cdot (\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) \\
h_6 &= (\eta_{20} - \eta_{02})((\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) + 4 \cdot (\eta_{30} + -\eta_{12})(\eta_{21} + \eta_{03}) \\
h_7 &= (3 \cdot \eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})(3 \cdot (\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) - \\
&\quad - (\eta_{30} - 3 \cdot \eta_{12})(\eta_{21} + \eta_{03})(3 \cdot (\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2)
\end{aligned} \tag{5.10}$$

OpenCV: Knihovna nabízí funkce *moments(...)* a *HuMoments(...)* pro výpočet momentových invariantů až do 3. řádu a Hu momentových invariantů. Pro srovnání dvou tvarů je k dispozici funkce *matchShapes(..)*, která s využitím Hu invariantů stanoví míru podobnosti dvou kontur.[5]

Srovnání kontur

Porovnání tvaru dvou objektu na základě kontur je možné provést s využitím Hu momentových invariantů. Jedná se tedy o sedm čísel popisujících tvar daného objektu. S využitím funkce *matchShapes(...)* je možné stanovit míru shody kontur dvou objektů. K dispozici jsou tři metody, které mají odlišné metriky porovnání (vztahy 5.11). Výstupem srovnávání je vždy jedno číslo, které podle zvolené metody (I_1, I_2, I_3) vyjadřuje podobnost kontur.

$$I_1(A,B) = \sum_{i=1}^{7} |\frac{1}{m_i^A} - \frac{1}{m_i^B}|$$

$$I_2(A,B) = \sum_{i=1}^{7} |m_i^A - m_i^B|$$

$$I_3(A,B) = \sum_{i=1}^{7} |\frac{m_i^A - m_i^B}{m_i^A}| \tag{5.11}$$

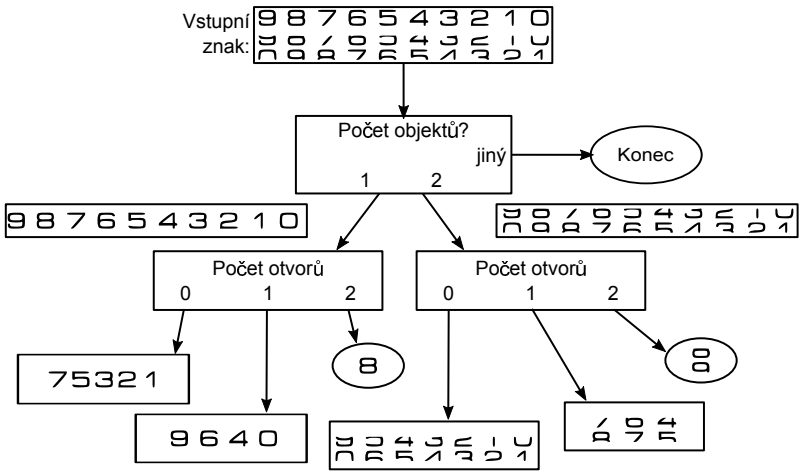
kde $m_i^A = sign(h_i^A) \cdot log|h_i^A|$; $m_i^B = sign(h_i^B) \cdot log|h_i^B|$; A, b - označení prvního a druhého obrazu, h_i Hu moment i-tého řádu.

Klasifikace

Než bude použito srovnávání kontur vzoru s testovacím obrazem, bude test zařazen do podmnožiny vzorů na základě počtu objektů a otvorů, které jsou jednoduše získatelné a mají velmi dobrou rozlišovací schopnost pro jednotlivé kategorie. K tomuto účelu byl vytvořen rozhodovací strom (obr. 5.18). Kořen stromu rozdělí množinu vzorů na 2 části, podle toho, jedná-li se o jednu celou číslici, či části dvou navazujících. Následně je provedeno dělení na základě počtu otvorů. V případě číslice 8 je klasifikace zde ukončena, protože má jako jediná mezi celými číslicemi dva otvory, obdobně přechod mezi 8 a 9. Je třeba upozornit na symbol přechodu číslic 4 a 5, ty jsou zařazeny jak v kategorii 0, tak 1 otvor. Může nastat situace, kdy otočení číselníku číslici 4 na hranici okna symbolu uřízne a tím se otvor ztrácí.
Jakmile je testovaný znak zařazený do podmnožiny vzorů a nebyla ukončena klasifikace, může být na základě srovnání Hu momentů stanovena hodnota.

Úspěšnost klasifikace

S využitím souboru testovacích dat (40 snímků elektroměrů), byla vytvořena množina pro testování rozpoznání znaků (236 číslic). Výsledek je uveden v tabulce 5.3. Tato metoda nedosáhla moc vysoké úspěšnosti, důvodem může být velká citlivost na deformace číslic způsobené osvětlením, nečistotami a otočením znaku.



Obr. 5.18: Rozhodovací strom pro rozčlenění znaků na menší skupiny dle počtu objektů a otvorů.

Metoda	Celkový počet znaků	Správně vyhodnocené znaky	Úspěšnost klasifikace [%]
Geometrické deskriptory	236	158	66,95

Tab. 5.3: Výsledky testů klasifikace s využitím geometrických deskriptorů.

5.3.2 Srovnávání významných bodů

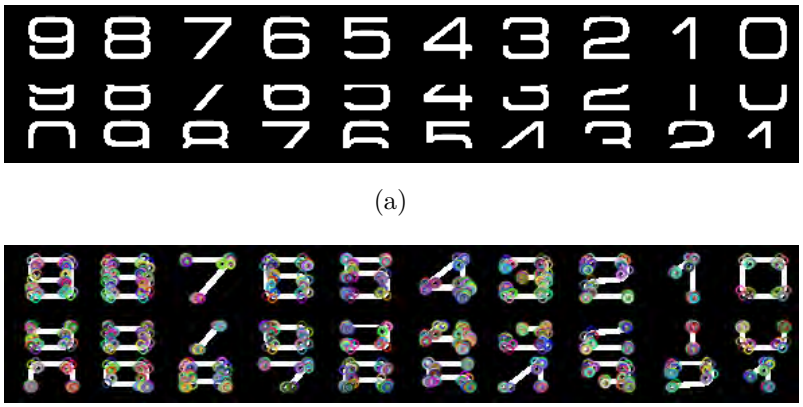
Významné body (Feature points, někdy označované jako příznaky) v obraze jsou takové body v obrazu, jejichž lokální okolí je určitým způsobem zajímavé. Předmětem zájmu můžou být hrany, rohy, či samostatné body. K lokalizaci takových bodů existuje celá řada detektorů. V případě rozpoznávání znaků budou předmětem zájmu rohy objektů v obrazu. Cílem je nalezení významných bodů v testovaném a vzorových obrazech s následným určení shody mezi nimi. Odpovídající si množiny bodů jsou určovány na základě jejich vlastností a relativní polohy bodů v množině. [4]

OpenCV: Pro detekci významných bodů je k dispozici implementace celé řady metod: Harris, Shi-Tomasi, SIFT (Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features), FAST (Features from Accelerated Segment Test), BRIEF (Binary Robust Independent Elementary Features), ORB (Oriented FAST and Rotated BRIEF).

Detekce

Cílem je najít v binárním obrazu jednoho znaku elektroměru velké množství významných bodů, aby v podstatě samy o sobě tvořily tvar číslice. Pro takový účel je

vhodným detektorem FAST (Features from Accelerated Segment Test). Tato metoda je navržena pro rychlou detekci bodů označujících rohy. Pro každý pixel je kontrolován okruh bodů (obvykle 16), pokud je nalezen na této kružnici nepřerušený úsek dvanácti bodů vyhodnocených jako světlé (oproti středovému pixelu), jedná se o roh. V obrazu obvykle není příliš velké množství rohů, ve srovnání s celkovým počtem pixelů, takže je třeba vyloučit velké množství oblastí a kontrola 16 bodů pro každí pixel je časově náročná. Z tohoto důvodu je v první fázi provedena rychlá verze, kde se kontrolují jen 4 protilehlé body. Příznaky, které projdou prvním testem, jsou následně kontrolovány v celém okruhu okolí. Tento postup často způsobí označení stejného rohu několika příznaky blízko sebe. Pro snížení počtu někdy redundantních bodů je použita nemaximální suprese, tedy potlačení všech příznaků malého okolí, které nemají maximální sílu.
V případě rozpoznání čísla, je ovšem více bodů vítaných, takže se nemaximální suprese nepoužije. Výsledek vyhledávání příznaků na obrazu vzoru číslic je na obr. 5.19.



(a)

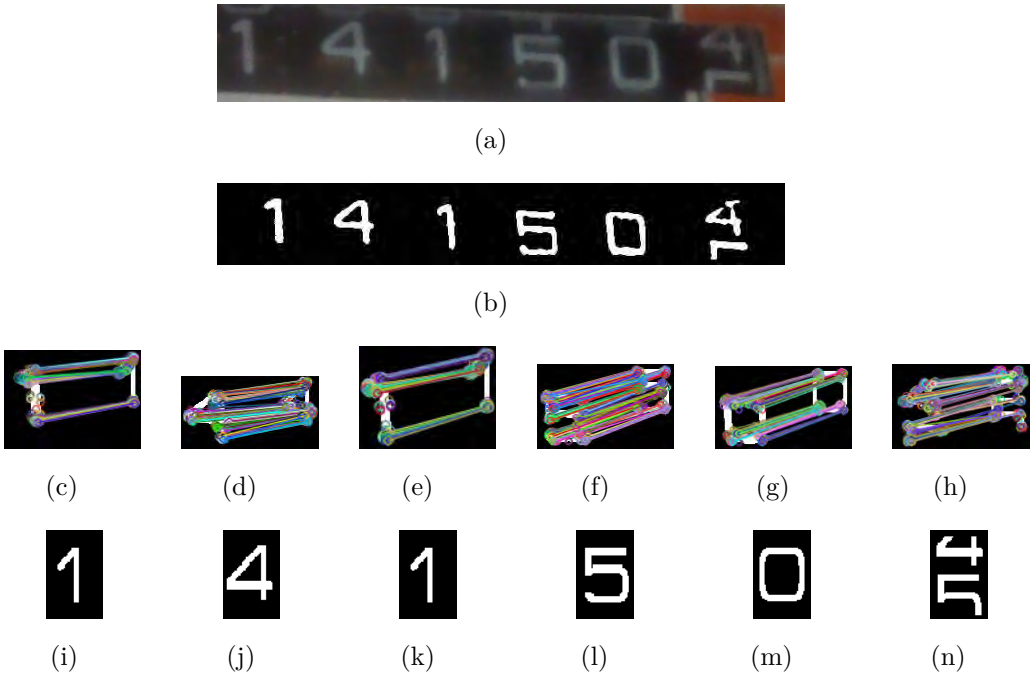
(b)

Obr. 5.19: Detekce významných bodů. (a) Vzory číslic elektroměru, (b) nalezené významné body metodou FAST.

Nalezení vzoru

Jakmile jsou nalezeny významné body jak v testovacím obrazu, tak ve všech vzorech, může se provést srovnávání. Pro každý nalezený příznak je extrahován jeho popis, ten záleží na použité metodě, ale může to být směr natočení, informace o jasových úrovních, síla příznaku a další. Nalezení shod mezi dvěma předloženými množinami významných bodů znamená pro každý příznak jedné množiny, projít všechny body druhé množiny a najít nejlepší shodu. Tento postup je označován jako hrubá síla (Brute-Force).
Příklad výsledku srovnávání číslic extrahovaných z číselníku je uveden na obrázku

5.20. Pro každou číslici byly nalezeny významné body, stejně jako pro všechny vzory (obr. 5.19). U každé testované číslice jsou vyhledány shodné významné body s každým vzorem. Vzor s největším počtem shodných bodů s testovanými udává výsledek.



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l) (m) (n)

Obr. 5.20: Rozpoznání číslic pomocí srovnávání se vzory. (a) Vstupní obraz, (b) Segmentované symboly, (c-h) nejlepší nalezené shody se vzory jednotlivých číslic (vlevo test, vpravo vzor), (i-n) rozpoznané vzory.

Tato metoda funguje velmi dobře i v případě rozdílných velikostí a natočení symbolů vzorů vůči testu. Výhodou je také necitlivost na další objekty v obraze. Na druhou stranu nastává problém s rozlišením číslic podobného tvaru, jako jsou 5, 6, 8 a 9. Z obrazu 5.19 je vidět, že použitý font způsobuje vnější kontury takových číslic příliš podobné a testovaná číslice může být chybně klasifikována. Potlačení tohoto jevu je možné pomocí rozčlenění vzorů do skupin podle jiných parametrů, např. podle geometrie jak bylo popsáno v kapitole 5.3.1.

Úspěšnost klasifikace

Metoda byla testována, obdobně jako předcházející, na množině 236 testovacích znaků. Použití srovnání významných bodů zlepšilo přesnost klasifikace oproti předešlé metodě (kapitola 5.3.1) o více než 10 %. Když byl aplikován rozhodovací strom snižující počet přípustných vzorů testovaného souboru, tak přesnost ještě vzrostla (tab. 5.4).

Metoda	Celkový počet znaků	Správně vyhodnocené znaky	Úspěšnost klasifikace [%]
Pouze významné body	236	187	79,24
Omezení počtu vzorů geometrií	236	207	87,71

Tab. 5.4: Výsledky testů klasifikace s využitím srovnání významných bodů.

5.3.3 Umělé neuronové sítě

Princip funkce umělých neuronových sítí (ANN - Artificial Neural Networks) je inspirován biologickými neuronovými sítěmi, které používají živá stvoření na Zemi. ANN je paralelní systém složený z jednodušších prvků (neuronů), jenž jsou mezi sebou propojeny vazbami (váhy). Takový systém je schopen požadovaného zpracování informací na základě změny vah vnitřních vazeb. Existuje více druhů ANN, které se liší různými parametry. Hlavní parametry definující ANN jsou:

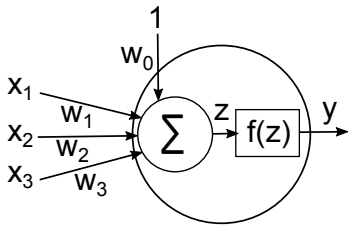
- Model neuronu - jedná se o základní stavební prvek ANN.
- Topologie - určuje způsob propojení jednotlivých neuronů (přímé vazby, zpětnovazební).
- Typ učení - hlavní dvě kategorie: jsou učení s učitelem a učení bez učitele.
- Vybavování - způsob určení výstupu naučené sítě v reakci na vstupní data.

Nejprve je třeba určit jaký typ sítě bude použit k účelu klasifikace číslic elektroměru. Na základě rešerše možností OCR [8], byla zvolena vícevrstvá síť perceptronů s algoritmem učení Back propagation. V následujících podkapitolách bude popsána právě tato ANN a její aplikace na problém rozpoznání číslic elektroměru.

Neuron

Základním stavebním prvkem takových sítí je neuron, který je zjednodušeným matematickým modelem biologického neuronu. Obsahuje n vstupů, které jsou přes příslušné váhy w přivedeny na vstup neuronu, stejně tak je přiveden takzvaný práh neuronu (bias) přes w_0. Vnitřní stav neuronu z je dán sumou vstupů. Výstupní hodnota neuronu je dána přenosovou funkcí $f(z)$, standardně se používá sigmoida. Vztah pro výpočet neuronu tedy dán vztahem 5.12. Takto popsaný neuron se označuje jako perceptron.

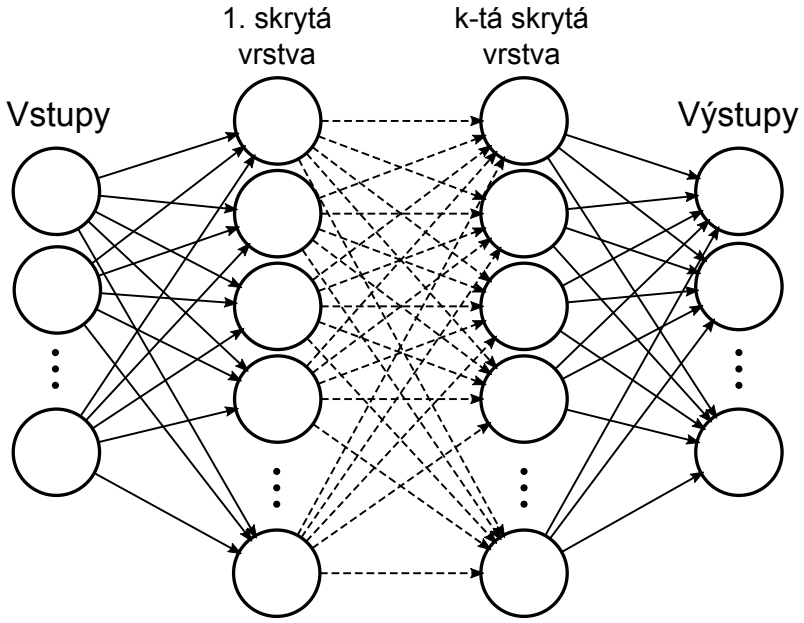
$$y = f(w_0 + \sum_{i=0}^{n} x_i w_i) \tag{5.12}$$



Obr. 5.21: Model neuronu.

Topologie

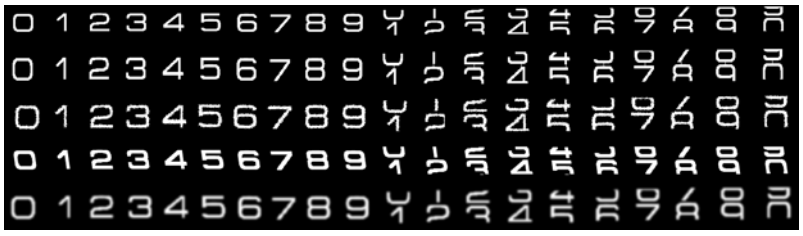
Jak bylo zmíněno výše, byla použita vícevrstvá síť perceptronů. Jedná se o síť bez zpětných vazeb, kde je mezi vstupní a výstupní vrstvou ještě k vnitřních vrstev. Na oprázku 5.22 je schématické zapojení této sítě. Je třeba stanovit počty jednotlivých neuronů. Množství neuronů vstupní vrstvy je určeno ze vstupních dat. Těmi jsou jednotlivé hodnoty pixelů binárního obrazu číslice, který byl normalizován na rozměr 30x30 pixelů. Tato velikost byla stanovena jako optimální hodnota dostatečně malá a zároveň bez viditelných zkreslení tvaru číslic. Počet neuronů vstupní vrstvy je tedy 900. Výstupní vrstva je určena počtem tříd, do kterých budeme vstupy klasifikovat, což je 20 (viz. obr. 5.12). Zbývá stanovit počet vnitřních vrstev a jejich neuronů. Tento počet byl určen experimentálně. Postupně byly otestovány počty neuronů vnitřních vrstev: 20, 40, 60, 80, 100 a to pro jednu a dvě vnitřní vrstvy. Experimentální výsledky jsou uvedeny na závěr kapitoly v tabulce 5.5.



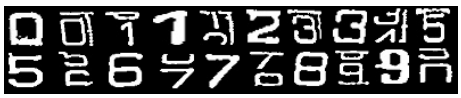
Obr. 5.22: Topologie dopředné vícevrstvé neuronové sítě perceptronů.

Trénování ANN

V případě vícevrstvé sítě perceptronů byl použit algoritmus učení Back propagation of error (zpětné šíření chyby). Jedná se o gradientní optimalizační metodu nastavení jednotlivých vah neuronové sítě vyhledáváním minima v multidimenzionálním prostoru váh. Učení probíhá sekvenčně pro jednotlivé tréningové vzory. Je důležité předložit v rámci učení ANN dostatečný počet vzorů z každé požadované třídy.
Tréningové vzory - v případě číslic elektroměru bude ANN klasifikovat do dvaceti tříd dle počtu možných znaků (číslice 0-9 a jejich přechody). Tréningová data byla vytvořena ze dvou zdrojů. Část byla pomocí již vytvořených metod extrahována ze snímků elektroměrů, druhá část byla vytvořena uměle. Nejprve byl nalezen font, který odpovídal tomu, co je používaný na číselnících elektroměru. S použitím tohoto fontu byla vytvořena první sada číslic. Z této se pomocí různých deformací (zúžení, roztáhnutí, dilatace, eroze, zkosení, rozpřepení okrajů) vytvořily další sady. Pak s pomocím testovacího software, který byl v rámci práce vytvořen (kapitola 4) byly jednotlivé znaky odděleny od sebe a normalizovány (30x30 pixelů) na velikost vhodnou pro ANN. Příklady vytvořených vzorů jsou na obrázku 5.23.



(a)



(b)

Obr. 5.23: Příklady tréningových vzorů ANN. (a) Uměle vytvořené vzory, (b) vzory extrahované ze snímků elektroměrů.

Vybavování

Jakmile je ANN naučena, všechny váhy jsou nastaveny tak, aby po předložení vzoru dané třídy výstup odpovídal této třídě. ANN může být použita ke klasifikaci číslic. Předložení vzoru znamená nastavení vstupů neuronové sítě podle hodnot vstupního obrazu číslice. Následně se od vstupní, přes vnitřní, až po výstupní vrstvy vyčíslí výstupy neuronů (vztah 5.12). Nejvyšší hodnota výstupu určuje příslušnou třídu.

OpenCV: Knihovna obsahuje modul ML(Machine Learning), který implementuje řadu tříd určených pro klasifikaci a shlukovou analýzu. Zde se nachází třída *CvANN_MLP*, která poskytuje vše potřebné pro tvorbu vícevrstvé neuronové sítě perceptronů. Výhodou je možnost exportu naučené ANN do souboru, který je možné následně načíst.

Úspěšnost

Vyhodnocení přesnosti ANN se provádí na testovací množině dat, která nebyla použita při učení. Na základě správného, či špatného přiřazení testovaného vzoru je stanovena úspěšnost v procentech. Testovací množina použitá v této práci se skládala ze 40 snímků elektroměrů, ze kterých byly segmentovány a normalizovány číslice. Výsledný počet testovaných číslic byl 236 (v některých případech selhala segmentace).Tyto číslice nebyly použity při učení ANN, aby bylo určení přesnosti nezávislé. Bylo otestováno deset naučených neuronových sítích (stejná tréningová množina) lišících se v počtu vnitřních vrstev a počtu jejich neuronů. V tabulce 5.5 jsou uvedeny výsledky testování jednotlivých ANN. Uveden je také počet iterací při učení, který může být, stejně jako úspěšnost, použit k stanovení vhodné topologie sítě. V případě sítě s dvěma vnitřními vrstvami o 20 neuronech došlo při učení k saturaci a při vybavování měla nezávisle na vstupech konstantní hodnotu. Optimální topologie byla stanovena na základě nejlepší úspěšnosti, v tabulce 5.5 je vyznačen příslušný řádek zeleně.

Počet vnitřních vrstev	Neuronů ve vnitřní vrstvě	Počet iterací při učení	Úspěšnost klasifikace [%]
1	20	383	73,73
1	40	346	85,59
1	60	355	71,19
1	80	327	87,29
1	100	714	80,93
2	20	2454	Saturace
2	**40**	**256**	**89,41**
2	60	294	63,38
2	80	771	80,28
2	100	430	78,87

Tab. 5.5: Výsledky testů ANN pro 1 a 2 vnitřní vrstvy a rozsah počtu vnitřních neuronů 20-100.

Implementace: Pro metodu geometrických deskriptorů a porovnávání významných bodů jsou vytvořeny samostatné třídy C++: *mycvDigitProcGeometric* a *mycvDigitProcFeatures*. Pro ANN byla nejprve vytvořena C++ implementace, která

je určená pro testovací software *mycvDigitProcANN* a implementuje proces učení. V Android aplikaci je ANN používána v Java kódu, podobně jako u kaskádového klasifikátoru je jednodušší přístup k souboru s naučenou sítí.

5.3.4 Srovnání metod OCR

Navrženy byly 3 metody, z nichž jedna (geometrické deskriptory) měla výrazně nižší úspěšnost rozpoznání znaků. Byla ovšem použita v kombinaci s metodou srovnávání významných bodů a v tomto případě je přesnost srovnatelná s poslední metodou využívající neuronové sítě.

5.4 Vyhodnocení

V tabulce 5.6 jsou shrnuty přesnosti jednotlivých segmentačních a klasifikačních metod použitých pro odečet hodnoty elektroměru. Pokud zvolíme sérii nejúspěšnějších metod segmentace (vyznačeno tučně) a klasifikace je výsledná úspěšnost rozpoznání dána násobkem dílčích. Pro pravděpodobnost správné klasifikace všech šesti číslic elektroměru platí:

$$0,975 \cdot 0,9631^6 \cdot 0,8941^6 \cdot 100 = \mathbf{39,75}\%$$

Vzhledem k této hodnotě byla v aplikaci Elektroměry zavedena možnost rychlé korektury (viz. obr. B.1), protože chyba nastane např. v jednom, či dvou znacích, uživatel může rychle chybu opravit.

Segmentace číselníku		Segmentace znaků		Rozpoznání znaků	
Metoda	Úspěšnost [%]	Metoda	Úspěšnost [%]	Metoda	Úspěšnost [%]
Determ.	97,5	Vert. Projekce	92,19	Geom. deskriptory	66,95
Haar	90	**Kontury**	96,31	Významné body	87,71
				Neuronová síť	89,41

Tab. 5.6: Souhrn výsledků testů jednotlivých metod zpracování.

6 ZÁVĚR

V úvodu práce je uveden popis a rozdělení elektroměrů se zaměřením na analogové typy, které jsou předmětem zájmu. Následuje první tématický celek, kterým je návrh mobilní aplikace pro systém Android. Je uveden popis tohoto systému a možnosti vývojových nástrojů, pro účely této práce bylo zvoleno vývojové prostředí Eclipse. Dále je popsán návrh aplikace, jejího grafického rozhraní a jednotlivých implementovaných tříd. Byla zajištěna kompatibilita aplikace s převažující většinou dnes používaných verzí Android OS v mobilních zařízeních. Praktické testy byly provedeny s přístroji: Samsung Galaxi Mini (Android 2.3.4), Lenovo A789 (Android 4.0.2), LG L Fino (Android 4.4.2). Při snímání kamerou zařízení byl rozsah rychlostí snímků 2,5 až 10 FPS.

Druhým tématickým celkem je zpracování obrazu elektroměru za účelem odečtení hodnoty číselníku. Tento problém byl řešen pomocí sekvence tří na sebe navazujících kroků.

Prvním z nich je lokalizace číselníku v obrazu elektroměru. Byla navržena dvě řešení tohoto problému. První vychází z apriorních znalostí o vlastnostech hledaného objektu a s využitím prahování a shlukování vertikálních čar je provedena lokalizace. Druhá metoda využívá aparát strojového učení, na základě množiny pozitivních a negativních vzorů byl vytvořen kaskádový klasifikátor, jenž skenováním oblastí snímku elektroměru lokalizuje číselník. Úspěšnost nalezení číselníku ve video-sekvenci kamery je pro první metodu 97,5% a pro druhou 90%.

Další krok k úspěšnému odečtení hodnoty je segmentace jednotlivých znaků z výřezu počítadla. Opět byly navrženy dva přístupy řešení. Prvním je využití vertikální projekce binárního obrazu, kde oblasti nízkých úrovní oddělují jednotlivé symboly. Druhá metoda využívá vyhledávání a selekce kontur objektů. Úspěšnost metody vertikální projekce je 92,2% a selekce kontur 96,3%.

Posledním krokem je klasifikace jednotlivých znaků. Nejprve byl proveden rozbor možných zobrazených hodnot. Kromě číslic od 0 do 9 se na rotačním číselníku elektroměru mohou nacházet přechody mezi jednotlivými číslicemi. Z toho důvodu byla klasifikace prováděna do dvaceti tříd pro jednotlivé číslice a všechny přechody mezi nimi. Otestovány byly tři přístupy rozpoznání číslic. První využívá geometrických deskriptorů, na jejichž základě je klasifikace provedena rozhodovacím stromem. Druhá metoda je založena na porovnávání významných bodů. Poslední přístup využívá umělou neuronovou síť, která je trénována zčásti extrahovanými vzory z reálných snímků a z části uměle vytvořenými znaky. První metoda dosáhla úspěšnosti klasifikace pouze 66,9%, je velmi citlivá na zkreslení odlesky a poškozením panelu elektroměru. V případě porovnávání významných bodů je úspěšnost 79,2%, ovšem když byla použita kombinace prvních dvou metod, výsledná úspěšnost je 87,7%.

Umělá neuronová síť poskytla nejlepší výslednou klasifikaci znaku 89,4%.
V rámci vývoje algoritmů zpracování obrazu byl vytvořen testovací software pro PC, který umožňuje testování s jedním i sekvencí obrazů, zahrnuje nástroje pro přípravu vzorů a učení neuronové sítě a jednoduché statistické vyhodnocení. Výše popsané metody navazují na sebe, takže jejich spojením lze dosáhnout různých výsledků. Pokud použijeme pro dílčí kroky metody vykazující nejvyšší úspěšnost bude pravděpodobnost správné klasifikace všech šesti číslic elektroměru 39,7%. I když je to poměrně malé číslo, není aplikace zbytečná. V grafickém rozhraní je při zobrazení odečtené hodnoty možnost rychlé korektury ze strany uživatele. V takovém případě je komfortně opravena jedna, či dvě chybně rozpoznané číslice a výsledek může být zařazen do databáze. Uložené hodnoty je možné graficky zobrazit v závislosti na čase.
Celkem byly splněny všechny požadavky zadání a nad rámec byl vytvořen software určený k testování algoritmů zpracování obrazu.

LITERATURA

[1] Activity. In: Android developer [online]. 2007 [cit. 2015-01-09]. Dostupné z: http://developer.android.com/reference/android/app/Activity.html

[2] Android (operating system). In: Android (operating system) [online]. 2014 [cit. 2014-12-25]. Dostupné z: http://en.wikipedia.org/wiki/Android_(operating_system)

[3] AndroidPlot [online]. [cit. 2015-05-13]. Dostupné z: http://androidplot.com/

[4] BAGGIO, Daniel Lélis. *Mastering OpenCV with practical computer vision projects*. 1. publ. Birmington: Packt Pub, 2012, 284 s. ISBN 978-184-9517-829.

[5] BRADSKI, Gary R. *Learning OpenCV*. Sebastopol: O'Reilly, c2008, xvii, 555 s. ISBN 978-0-596-51613-0.

[6] DENIZ SUAREZ, Oscar. OpenCV Essentials. Birmingham, UK: Packt Pub., 2014, 214 s. ISBN 978-1-78398-424-4.

[7] Elektrotechnická měření 1. vyd. Praha: BEN - technická literatura, 2002, 255 s. ISBN 80-730-0022-9.

[8] EUGENE, Borovikov. 2004. A survey of modern optical character recognition techniques [online]. : 36 [cit. 2015-05-12]. Dostupné z: http://www.academia.edu/3732087/A_survey_of_modern_optical_character_recognition_techniques

[9] FELKER, Donn a Joshua DOBBS. *Android application development for dummies*. Hoboken, NJ: Wiley Pub., c2011, xx, 357 p. –For dummies. ISBN 04-707-7018-X.

[10] FLUSSER, Jan, Tomáš SUK a Barbara ZITOVÁ. 2009. Moments and moment invariants in pattern recognition. 1st pub. Chichester: John Wiley and Sons, xiv, 296 s. ISBN 978-0-470-69987-4.

[11] HORÁK, Karel. Matematická morfologie. In: [online]. [cit. 2015-04-30]. Dostupné z: http://midas.uamt.feec.vutbr.cz/POV/lectures-pdf/08_Matematicka_morfologie.pdf

[12] Hough Line Transform. In: [online]. opencv dev team, Feb 25, 2015 [cit. 2015-04-30]. Dostupné z: http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html

[13] HOWSE, Joseph. *Android application programming with openCV: build Android apps to capture, manipulate, and track objects in 2D and 3D*. Birmingham: Packt Publishing, c2013, 115 s. ISBN 978-184-9695-206.

[14] HU, Ming-Kuei. 1962. Visual pattern recognition by moment invariants. IEEE Transactions on Information Theory. 8(2): 179-187. DOI: 10.1109/tit.1962.1057692.

[15] Jednofázové elektromechanické elektromery. In: [online]. Elektroměry s.r.o., 09-2001 [cit. 2015-05-02]. Dostupné z: `http://www.elektromery.com/soubory/EJsl.pdf`

[16] KALOVÁ, Ilona. Segmentace a detekce geometrických primitiv. In: [online]. [cit. 2015-04-30]. Dostupné z: `http://midas.uamt.feec.vutbr.cz/POV/lectures-pdf/05_Segmentace_a_detekce_geometrickych_primitiv.pdf`

[17] KOO, Keunhwi, Jong PIL YUN, SungHoo CHOI, JongHyun CHOI, Doo CHUL CHOI a Sang WOO KIM. Recent advances in signal processing , robotics and automation: proceedings of the 8th WSEAS International Conference on Signal Processing, Robotics and Automation, Cambridge, UK, February 21-23, 2009. S. l.: WSEAS Press, 2009, s. 293-298. ISBN 9789604740543.

[18] SAPLIN, Maxim. Bilateral image filter: Edge preserving blur and noise reduction. In: [online]. 2012 [cit. 2015-04-29]. Dostupné z: `http://saplin.blogspot.cz/2012/01/bilateral-image-filter-edge-preserving.html`

[19] Security. In: *Open Source Project* [online]. [cit. 2015-01-09]. Dostupné z: `https://source.android.com/devices/tech/security/`

[20] SHI, Shin. 2013. Emgu CV essentials: develop your own computer vision application using the power of Emgu CV. Birmingham, UK: Packt Publishing. ISBN 978-178-3559-527.

[21] ŠONKA, Milan, Václav HLAVÁČ a Roger BOYLE. Image processing, analysis, and machine vision. 3rd ed. Toronto: Thomson, 2008, xxv, 829 s. ISBN 978-0-495-08252-1.

[22] Tegra Android Development Pack. In: *NVIDIA DEVELOPER ZONE* [online]. 2014 [cit. 2014-12-26]. Dostupné z: `https://developer.nvidia.com/tegra-android-development-pack`

[23] *The OpenCV Reference Manual* [online]. 2.4.8.2. [cit. 2015-01-08]. Dostupné z: `http://docs.opencv.org/opencv2refman.pdf`

[24] TouchImageView. 2011. MIKE, Ortiz. Github [online]. [cit. 2015-05-13]. Dostupné z: https://github.com/MikeOrtiz/TouchImageView

[25] VIOLA, P. a M. JONES. 2001. Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001 [online]. IEEE Comput. Soc, s. I-511-I-518 [cit. 2015-05-08]. DOI: 10.1109/CVPR.2001.990517. ISBN 0-7695-1272-0. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=990517

[26] Yungang Zhang, Changshui Zhang. A New Character Segmentation Algorithm for License Plate. The 2003 Intelligent Vehicles Symposium (IV2003), Columbus, OH, USA, June 9-11, 2003, Proceedings. pp.106 - 109. Dostupné z: http://bigeye.au.tsinghua.edu.cn/english/paper/A%20New%20Algorithm%20for%20Character%20Segmentation%20of%20License%20Plate.pdf

[27] 3F1T G1Y6. *Skupina ČEZ* [online]. [cit. 2014-12-25]. Dostupné z: http://www.cez.cz/cs/co-delat-kdyz/technicke-zalezitosti/pro-odberatele/pruvodce-elektromery/3f1tg1y6.html

[28] 3F2T G1Y6d. *Skupina ČEZ* [online]. [cit. 2014-12-25]. Dostupné z: http://www.cez.cz/cs/co-delat-kdyz/technicke-zalezitosti/pro-odberatele/pruvodce-elektromery/3f2tg1y6d.html

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ADC	Analog to Digital Converter
ANN	Artificial Neural Networks
FAST	Features from Accelerated Segment Test
FPS	Frames Per Second
GUI	Graphical User Interface (Grafické uživatelské rozhraní)
HSV/HSL	Hue - barevný tón, Saturation - sytost barvy, Value/Lightness/Luminance - jasová složka
IDE	Integrated Development Environment
JNI	Java Native Interface
OAH	Open Handset Alliance
OCR	Optical Character Recognation
OS	Operační Systém
SDK	Software Development Kit
SDL	Simple DirectMedia Layer
SQL	Structured Query Language
XML	Extensible Markup Language
YUV	Y - jasová složka, U a V - barevné složky

SEZNAM PŘÍLOH

A Přiložené soubory **71**

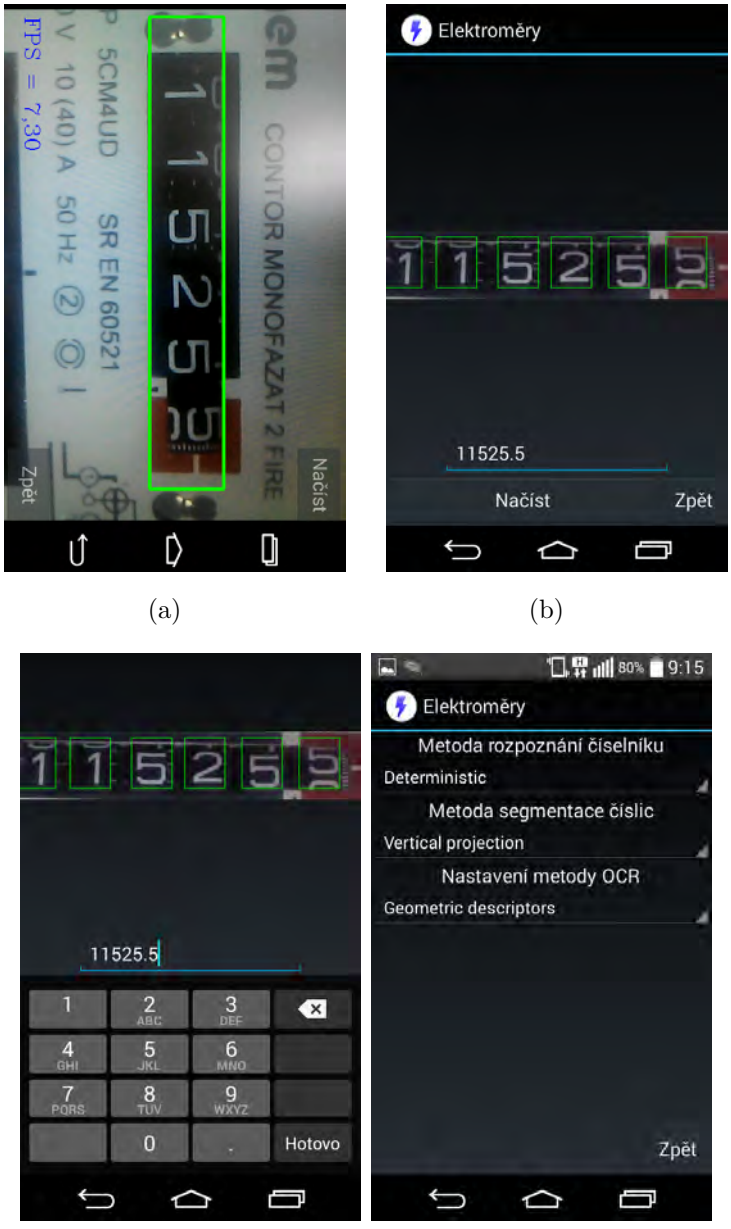
B Screen aplikace **72**

A PŘILOŽENÉ SOUBORY

V příloze se nachází následující soubory:

- Tabulka měření přesnosti.
- Projekt aplikace Elektroměry, release aplikace.
- Testovací software.

B SCREEN APLIKACE



(a) (b)

(c) (d)

Obr. B.1: Screeny aplikace.